Universidad Autónoma de Madrid

Escuela Politécnica Superior

Grado en Ingeniería Informática

TRABAJO DE FIN DE GRADO

DESARROLLO DE UNA APLICACIÓN DE CONFIGURACIÓN PARA SISTEMAS DE NAVEGACIÓN AUTÓNOMOS EN UAV

Cristina Ortega Lara
Tutor: Juan Manuel Luque Suárez
Ponente: Fernando Jesús López Colino

Julio 2016

DESARROLLO DE UNA APLICACIÓN DE CONFIGURACIÓN PARA SISTEMAS DE NAVEGACIÓN AUTÓNOMOS EN UAV

Autor: Cristina Ortega Lara
Tutor: Juan Manuel Luque Suárez
Ponente: Fernando Jesús López Colino

Departamento de Software
Escuela Politécnica Superior
Universidad Autónoma de Madrid

Julio 2016

Agradecimientos

Quisiera agradecer a varias personas la ayuda que me han prestado en la realización de este proyecto. Entre ellas, y en primer lugar, a David, por todo lo que me ha enseñado y transmitido en estos meses.

También quiero agradecer la ayuda de todo el equipo de UAV Navigation, incluidos mis apreciados becarios, Fidel y Sebas, dispuestos a ayudarme, aconsejarme y hacerme reír en los momentos de mayor tensión.

A mis amigas Sandra, Irene y Elena por escucharme y darme ánimos cuando la situación me desbordaba, a mi mejor amigo Antonio por colorear los momentos grises, a Roberto y Guido, por sus peculiares cafés, y a Julián por estos cuatro años de delirios en la EPS.

Le agradezco también a José María la paciencia que ha tenido conmigo, la fuerza que me ha transmitido, los consejos que me ha dado y los inolvidables y geniales momentos que hemos vivido, los cuales han hecho que el desarrollo de este proyecto haya sido más ameno a lo largo de estos meses.

A toda mi familia: a mi hermana, que me ha cuidado como nadie en los peores momentos, a mi madre, que sin ella me habría derrumbado hace tiempo y por último a mi padre, quien no ha dudado en ningún momento de mi valía y siempre me ha apoyado en todo lo que he decidido.

Y por último, al más importante, a mi gran amigo Roberto Mora Lara, por descubrirme este mundo de cables y programas, y por ser la mejor persona y el mejor amigo que alguien puede tener.

"Quien algo quiere, algo le cuesta"

Abstract

Abstract — The Unmanned Aerial Vehicle (UAV), commonly known as drones, are unmanned aerial vehicles, equipped with highly innovative systems like Global Positioning System (GPS), sensors, cameras, radar controls, etc. capable of sending data just in milliseconds to the earth station from where they are controlled. Now there are different current projects about the use of drones, for example, for the dispensing of traded goods, in Amazon case, the vigilance of big and dangerous tracts of land, like the Turrialba volcano, or even in attacks against human targets, in the case of USA intelligence services. All missions added in these devices need an accurate preparation and control of themselves.

On the basis of these principles, and as a result of this End of Degree Project, it is given an application that makes easier the UAV's configuration from the earth station, Visionair.

During the project's development process, there have been utilities incorporated that increased the UAV's parameters range that could be configured. For these utilities development, it has been conducted a thorough requirement analysisthat covered all the characteristics the application needed. Then, it has been made a design in which it has been created different profiles for the different devices. These profiles contained the utilities that each device needed for them component configuration and, in this way, provide the user an application which is oriented to this particular device in order to minimize the existence of useless utilities.

The design has overcome successfully the Acceptance Test Plan (ATP) it was tested and soon it will be available so the clients and the UAV Navigation Company engineers have an integrated tool in Visionair. This tool allows engineers to configure their devices so that they don't have to use any kind of external applications.

Key words — UAV, Drones, Earth station, Visionair, ATP.

Resumen

Resumen — Los UAV, más conocidos como drones, son vehículos aéreos no tripulados, equipados con sistemas de última generación como GPS, sensores, cámaras, control de radares, etc. capaces de enviar información en milisegundos, a la estación de tierra desde la que son controlados. En la actualidad, existen diversos proyectos en marcha sobre el uso de drones, por ejemplo, para la dispensación de productos comerciales, en el caso de Amazon; la vigilancia de grandes y peligrosas extensiones de tierra, como el volcán de Turrialba; o incluso en ataques contra blancos humanos, en el caso de los servicios de inteligencia de los Estados Unidos. Todas las misiones que incorporan estos dispositivos precisan de una rigurosa preparación y control de los mismos.

En base a estos principios, como resultado de este Trabajo de Fin de Grado se presenta una aplicación que facilita la configuración de los UAV desde la estación de tierra, Visionair.

Durante el proceso de desarrollo del proyecto, se han ido incorporando utilidades que ampliaban el abanico de parámetros del UAV que se podían configurar. Para el desarrollo de estas utilidades, se ha llevado a cabo un exhaustivo análisis de requisitos que albergaba todas las características y necesidades que debía cubrir la aplicación. A continuación, se ha realizado un diseño en el que se han creado distintos perfiles para los diferentes dispositivos. Estos perfiles contenían las utilidades que cada aparato precisaba para la configuración de sus componentes, para así poder ofrecer al usuario una aplicación orientada a la configuración de ese dispositivo concreto y, de esta manera, reducir al máximo la existencia de utilidades inservibles.

El diseño ha superado con éxito el ATP al que se le ha sometido y, muy pronto, estará disponible para que tanto los clientes como los ingenieros de la empresa UAV Navigation puedan contar con una herramienta integrada en Visionair que les permitirá configurar sus dispositivos sin necesidad de aplicaciones externas.

Palabras clave — UAV, Drones, Estación de tierra, Visionair, ATP.

Glosario

AP04 Piloto automático plenamente integrado que ha estado en servicio desde 2004. Actualmente, ha sido sustituido por el dispositivo VECTOR, mejorado y con más prestaciones. 6, 10, 11, 14, 19, 23, 31, 32, 34, 35, 53, 58, 63, 69

IDLE Estado de reposo. 14

Knowledge Base Página web que contiene la información sobre los productos de UAV Navigation. 18

POLAR Es un dispositivo que tiene integrado un Air Data Attitude Heading Reference System (ADAHRS) y un Inertial Navigation System (INR) con un GPS asistido. 11, 12, 14, 38, 52

UAV Navigation Empresa privada especializada en el desarrollo de pilotos automáticos, sistemas de control de vuelo y módulos de procesamiento de movimiento utilizados en dispositivos UAV. III, V, 1–6, 9, 10, 20, 23–25

VECTOR Piloto automático totalmente integrado para su uso en UAV, donde se requiere el máximo rendimiento. 10, 11, 14, 19, 23, 31, 32, 34, 35, 52, 58, 64, 70

Visionair Software estándar de la estación de tierra perteneciente a la empresa UAV Navigation y utilizado para la planificación y ejecución de misiones con UAV. III, V, 2, 3, 9, 10, 17–21, 23–25, 28, 34, 37

Acrónimos

ACK Acknoleadge. 21, 28, 30

ADC Analog-to-Digital Converter. 13, 55

ATP Acceptance Test Plan. III, V, 18

CAN Controller Area Network. 14

COM COMmunications. 14, 16, 18

CPU Central Processing Unit. 10, 23, 24, 36

CRC Cyclic Redundancy Code. 11, 51

DLL Dynamic Link Library. 9, 18, 20, 25, 37

GCS Ground Control Station. 38

GPS Global Positioning System. III, V, 16

ICAO International Civil Aviation Organization. 16

ICD Interface Control Documentation. 2, 9, 20, 33, 37

ID Identification Data. 16, 20

IP Internet Protocol. 13, 31, 53, 68

LINK Logistics Information Network. 14

MUAV Micro Unmanned Aerial Vehicle. 5

MVVM Model View ViewModel. 21, 33

OAT Outside Air Temperature. 10

PGM Programmable. 12, 52

PWM Pulse Width Modulation. 13

RPAS Remotely Piloted Aircraft Systems. 6

TCP Transmission Control Protocol. 18

UAV Unmanned Aerial Vehicle. III, V, 1, 2, 6, 9, 12, 16, 25, 37

UDP User Datagram Protocol. 13, 18

WF Windows Form. 25

WPF Windows Presentation Foundation. 24, 25, 30, 37

XML Extensible Markup Language. 10, 11, 14, 27, 28, 34, 35, 53, 54, 58, 59

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

En este capítulo se expondrán los motivos que han llevado a la realización de este proyecto. A continuación, se especificarán los objetivos finales del proyecto, las fases en las que se ha dividido y, por último, se detallará la estructura del resto del documento.

1.1. Motivación

En la actualidad, cuando se habla de dispositivos UAV, tanto los diferentes componentes hardware como el software embarcado a bordo de este son elementos que deben ser tratados con mucho cuidado y consideración.

Es importante señalar que al hablar del hardware de un UAV nos estamos refiriendo tanto a los componentes que van ensamblados en la placa como a los diferentes componentes hardware que pueden ser conectados a ésta. Igualmente, al hablar de software, nos referimos tanto a los sistemas operativos que están embebidos en estos dispositivos, y que permiten su funcionamiento, como a los propios datos que este sistema operativo utiliza para funcionar o comunicarse con el exterior.

Las consecuencias de un fallo en un dispositivo de estas características pueden provocar efectos insignificantes, que no afectan a las prestaciones del dispositivo y sus sistemas, o efectos catastróficos, como un accidente que deje inoperativo el dispositivo, suponga una pérdida considerable de dinero o incluso provoque daños en humanos.

En la actualidad, la adecuada configuración de este tipo de dispositivos es muy importante antes, durante y después de su puesta en marcha. Para los trabajadores de la empresa UAV Navigation, es esencial que este tipo de configuración se realice de una forma precisa y robusta, de esta manera garantizan que sus productos sean de calidad y minimizan enormemente las probabilidades de fallo.

Por otro lado, encontramos un panorama en el que se lucha en busca de la usabilidad, cualidad que hacen mucho más atractivo un software. Es mucho más seductor un software que aglutina dos funcionalidades que convergen en un punto y cuyo uso es sencillo, que dos

aplicaciones que contienen una única funcionalidad, donde las acciones de una repercuten en la otra o simplemente guardan una estrecha relación.

Por lo tanto, se busca especificar un software concreto y determinado para los dispositivos de la empresa UAV Navigation que permita, tanto a los ingenieros como a los clientes, configurar los dispositivos que comercializan de una forma precisa y sencilla.

1.2. Objetivos

Estudiando el problema, el objetivo principal de este proyecto es el diseño de una aplicación que permita la configuración remota de todos los parámetros de un sistema de navegación autónomo y que éste corra dentro del software Visionair.

Para ello, se deberá hacer un estudio de los sistemas software de configuración de los UAV disponibles, de los protocolos de comunicación utilizados en los productos de UAV Navigation y de los requisitos que el software deberá contener y cumplir para dar ese servicio a ingenieros y clientes.

Debido a la necesidad de integración con Visionair, es preciso utilizar los mecanismos de comunicación y carga de este software y la creación de una librería de comunicaciones que abarque el Interface Control Documentation (ICD).

Por todo esto, los objetivos principales del proyecto son los siguientes:

- Desarrollo de una librería de comunicaciones que implemente el ICD de UAV Navigation.
- Desarrollo de una aplicación de configuración remota de parámetros de sistemas UAV.
- Integración de la aplicación de configuración en Visionair.

1.3. Fases de realización

Para el desarrollo de este proyecto, se han definido una serie de fases:

- **Estudio, análisis y documentación:** la primera fase se compuso de un análisis en profundidad del problema planteado: ¿qué se desea?, ¿qué necesidades debe cubrir? y ¿cuáles son las limitaciones? A continuación, se realizó un estudio de los recursos de los que se disponía, haciendo especial hincapié en la documentación, que recoge protocolos de comunicación y manuales de usuario, y dispositivos hardware necesarios para el desarrollo. Por último, se desarrolló un primer documento que recogía un resumen con las intenciones del proyecto y una detallada descripción de

los requisitos definidos en una primera reunión con los ingenieros de Flight Control y responsables de márquetin.

- **Revisión de la documentación:** en esta segunda fase, se envió el documento elaborado en la primera fase a los jefes de los distintos departamentos de UAV Navigation, buscando conformidad, nuevas incorporaciones de requisitos o eliminación de algunos ya obsoletos.

- **Preparación del entorno de trabajo:** durante la fase de preparación, tuvo lugar una reunión con el jefe de departamento de software para estudiar las diferentes opciones de desarrollo de la aplicación, entre las que se encontraba la definición del lenguaje de la aplicación, el entorno de trabajo, la estructura del proyecto y los posibles cambios necesarios en el software de Visionair para la integración de la nueva aplicación.

- **Diseño e implementación de la aplicación:** a lo largo de esta fase, realizó el diseño del proyecto y se desarrollaron tanto la librería de comunicaciones como la aplicación que dieron lugar a este proyecto.

- **Realización de pruebas:** en la fase de pruebas, se elaboró un plan de pruebas que recogía pruebas unitarias, de sistema e integración y pruebas de validación. También se corrigieron los errores encontrados tras pasar estas baterías de pruebas.

- **Redacción de la memoria:** por último, se terminó de desarrollar este documento que recogía todo el proceso de realización del proyecto de forma detallada.

A continuación, en la Figura 1.1, se muestra un diagrama de Gantt que plasma en una línea temporal la duración de las fases descritas anteriormente. Como se puede observar, sigue un ciclo de vida en cascada ya que ha sido preciso finalizar una fase del proyecto para comenzar la siguiente.

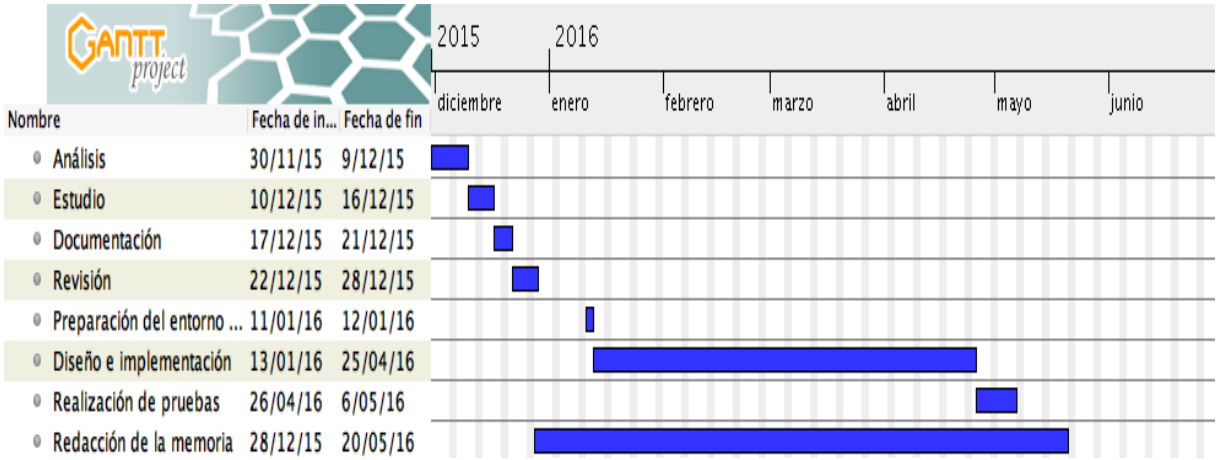


Figura 1.1: Diagrama de Gantt.

1.4. Estructura del documento

La memoria consta de los siguientes capítulos:

- **Capítulo 2**: en este capítulo se realizará un análisis del concepto de dron y de sus usos. Posteriormente, se hablará de la empresa UAV Navigation, empresa que comercializará el software nacido tras el desarrollo de este proyecto.
- **Capitulo 3**: recogerá el análisis del problema que conduce a la realización de este proyecto, detallando su alcance y los requisitos que ha tenido que cumplimentar el software.
- **Capítulo 4**: se especifica el diseño del proyecto de forma general y después profundizando en cada una de sus partes.
- **Capítulo 5**: contiene el desarrollo de cada uno de los componentes del proyecto.
- **Capítulo 6**: en él se detallan las pruebas realizadas y los resultados obtenidos de ellas.
- **Capítulo 7**: por último, se exponen las conclusiones del proyecto, si se han cumplido los objetivos marcados, los conocimientos adquiridos y los puntos que se abordarán en un trabajo futuro para ampliar la funcionalidad de la aplicación.

Capítulo 2

Estado del Arte

En esta sección se va a dar una pequeña definición del concepto de dron. Posteriormente, se hará un repaso sobre la actualidad de los drones y sus usos. Para finalizar, se presentará la empresa UAV Navigation, propietaria del software derivado de este proyecto.

2.1. Actualidad de los drones

Hoy en día, se definen los drones como vehículos aéreos capaces de volar sin tripulación.

Con la vertiginosa velocidad de evolución de la tecnología, estas aeronaves han ido incorporando sistemas cada vez más sofisticados y se han conseguido grandes avances.

Actualmente, estos dispositivos han logrado millones de usos como pueden ser [1]:

- **De blanco**: para simular aviones o ataques de enemigos en sistemas de defensa de tierra o aire. En la actualidad, los ejércitos de muchos países utilizan los drones con fines de entrenamiento.
- **Reconocimiento**: envío de información militar. Entre estos, destacan los Micro Unmanned Aerial Vehicle (MUAV). Un ejemplo de drones destinados a este tipo de misiones se encuentra en Rusia, donde el ejército ruso ha iniciado misiones de reconocimiento con drones sobre el territorio Sirio [2]. Otros ejemplos curiosos de su empleo en este campo son la reproducción de rutas de aves en libertad o el estudio de restos arqueológicos, por ejemplo, el estudio de las ruinas de Cerro Chepén en Perú [3].
- **Logística**: diseñados para llevar cargas. En Alemania, la empresa DHL pretende realizar determinadas entregas de sus paquetes, en menos de veinticuatro horas, mediante el uso de drones. En las pruebas realizadas, en una isla alemana de unas 2000 personas, consiguieron dispensar medicinas a sus habitantes en menos de media hora [4].

- **Investigación y desarrollo**: en ellos se prueban e investigan los nuevos sistemas en desarrollo.
- **UAV comerciales y civiles**: diseñados para propósitos civiles. Actualmente, estos dispositivos están al alcance de todos los bolsillos, por lo que muchos terminan convirtiendo el vuelo de estos aparatos en hobbies.

Todos estos avances han ampliado el número de consumidores de este tipo de dispositivos, desde los más pequeños hasta los más profesionales del sector.

Debido a esto, el auge de los drones en el mercado ha sido tal que cada vez hay un mayor número de empresas que emergen en este mercado. Una de estas compañías emergentes es la empresa UAV Navigation.

2.2. UAV Navigation

UAV Navigation es una empresa privada especializada en el desarrollo de pilotos automáticos, sistemas de control de vuelo y módulos de procesamiento de movimiento utilizados en sistemas Remotely Piloted Aircraft Systems (RPAS), también conocidos como UAV [5].

La empresa, pionera en el sector, ha conseguido grandes logros que le han permitido seguir creciendo y desarrollar nuevos proyectos. A continuación, se remarcan algunos de ellos.

En 2010, su piloto automático, AP04, dirigió el vuelo de todos los aviones blanco, Scrab, de las Fuerzas Armadas españolas, los cuales eran utilizados para el entrenamiento de unidades de defensa tierra-aire y aire-aire. El AP04 permitía el despegue y aterrizaje automático, así como la realización del plan de vuelo de forma autónoma. La decisión de escoger el AP04 vino dada por su precisión, sus reducidas dimensiones, el incremento de las capacidades que aporta y la capacidad de dirigir autónomamente todas las fases de vuelo [6].

Figura 2.1: Scrab.

Años más tarde, en 2013, la empresa era reconocida como compañía pionera en el desarrollo de sistemas de navegación inercial y electrónica aplicada a la aviación para aeronaves no tripuladas. Fue galardonada por su contribución al sector energético e industrial durante la “II Cumbre Spain Startup & Investor 2013”, punto de encuentro de inversores nacionales e internacionales de emprendedores españoles donde se dan a conocer las mejores iniciativas de los emprendedores españoles [7].

En 2015, esta empresa fue la primera en España en conseguir una operación de vuelo automático de un dron más allá de la línea del alcance visual. El dron, con un peso de 45 kilos, capaz de alcanzar velocidades de 130 kilómetros por hora, un alcance de 400 kilómetros y capaz de lograr los 3500 metros de altura, consiguió recorrer una distancia de 20 kilómetros [8].

Estos son algunos ejemplos de sus logros en el territorio español, sin embargo, cuentan con muchos más en otros países, como su colaboración en el proyecto Martin Jetpack en Nueva Zelanda [9].

Actualmente, la empresa cuenta con numerosos proyectos en diversos países y no paran de trabajar con el objetivo de seguir creciendo y desarrollar nuevos proyectos que les hagan ser líderes mundiales. Uno de estos proyectos consiste en la mejora de las herramientas que les permiten configurar sus autopilotos y es aquí donde nace el presente proyecto.

Capítulo 3

Análisis

A continuación, se procederá a hacer un análisis de los dos sistemas que se han implementado, la librería de comunicaciones y la aplicación, donde se detallarán los objetivos, el alcance y los requisitos de los mismos.

3.1. Definición del proyecto

3.1.1. Objetivos y funcionalidad

Como se ha explicado anteriormente, el objetivo principal de este proyecto es el desarrollo de una herramienta que permita la configuración de dispositivos UAV remotamente a usuarios de un hardware de UAV Navigation. Para conseguir tal propósito, ha sido necesaria la creación de dos herramientas que se complementasen.

La primera de ellas, y la más importante, se presenta en forma de librería de comunicaciones. Se trata de una librería que implementa el ICD, es decir, contiene los mecanismos necesarios para la descomposición de los mensajes que se reciben del UAV y la creación de nuevos mensajes para enviar al mismo, atendiendo a unos nuevos valores definidos por el usuario. Esta librería utiliza el driver de comunicaciones que posee Visionair, el cual será explicado en el Capítulo 4.

En segundo lugar, se ha desarrollado una aplicación que sirve al usuario para comunicarse con el dispositivo. Esta aplicación consiste en una interfaz gráfica que sirve de puente entre el usuario y la librería de comunicaciones. Desde esta interfaz, el usuario podrá ejecutar la orden para la modificación de los diversos parámetros del dispositivo o la visualización de otros.

Estos dos componentes han sido encapsulados en una Dynamic Link Library (DLL) que Visionair cargará en su inicio.

3.1.2. Alcance

La herramienta desarrollada no es un prototipo, sino que es una aplicación que será comercializada por la empresa UAV Navigation. La aplicación pretende comunicar información de configuración al dispositivo conectado a Visionair y también servir de base para la creación del resto de aplicaciones que conforman las User Tools, las cuales serán explicadas en la Sección 4.1. Únicamente se implementa el panel de usuario, que permite una configuración básica del autopiloto, no el de desarrollador, que permite una configuración más avanzada del dispositivo. Este último queda como trabajo futuro.

3.2. Análisis de requisitos

3.2.1. Requisitos funcionales

A continuación, se describen los requisitos funcionales de cada una de las pestañas de la aplicación. Cabe destacar que, en determinadas pestañas, existen requisitos diferentes para los dispositivos VECTOR y AP04.

RF 1. Pestaña Basic.

- **RF 1.1.** El usuario podrá visualizar una brújula que muestra la posición del dispositivo.
- **RF 1.2.** El usuario será capaz de visualizar el número de serie del dispositivo.
- **RF 1.3.** El usuario será capaz de visualizar la temperatura del dispositivo.
- **RF 1.4.** El usuario será capaz de visualizar la Central Processing Unit (CPU) que en ese momento utiliza el dispositivo.
- **RF 1.5.** El usuario será capaz de visualizar el porcentaje usado de la CPU.
- **RF 1.6.** El usuario será capaz de visualizar la temperatura externa (Outside Air Temperature (OAT)).
- **RF 1.7.** El usuario podrá activar/desactivar un dispositivo mediante la radio dado un número de serie.
- **RF 1.8.** El usuario podrá activar/desactivar todos los dispositivos que tengan la misma configuración de radio.

RF 2. Pestaña Aircraft.

- **RF 2.1.** La configuración actual de un dispositivo podrá ser almacenada en un fichero Extensible Markup Language (XML).
- **RF 2.2.** La configuración de un dispositivo podrá ser cargada desde un fichero XML.
- **RF 2.3.** La configuración actual de un dispositivo podrá ser descargada.

RF 3. Pestaña Servos.

Requisitos comunes.

RF 3.1. Las opciones de los desplegables se corresponderán con las opciones del archivo "Servos.xml" ubicado en la carpeta "Documents/Visionair/UserTools".

RF 3.2. El usuario será capaz de visualizar y modificar el valor mínimo del servo.

RF 3.3. El usuario será capaz de visualizar y modificar el valor máximo del servo.

RF 3.4. El usuario será capaz de visualizar y modificar el valor central del servo.

RF 3.5. El usuario será capaz de visualizar y modificar la inversión del movimiento del servo.

RF 3.6. El usuario será capaz de asignar un canal específico a cada servo.

RF 3.7. La configuración actual de un dispositivo podrá ser almacenada en un fichero XML.

RF 3.8. La configuración de un dispositivo podrá ser cargada desde un fichero XML.

RF 3.9. La configuración actual de un dispositivo podrá ser descargada.

Requisitos para VECTOR.

RF 3.10. El usuario será capaz de visualizar y modificar el valor mínimo comandado del servo.

RF 3.11. El usuario será capaz de visualizar y modificar el valor máximo comandado del servo.

Requisitos para AP04.

RF 3.12. El usuario será capaz de visualizar y modificar el rango del servo.

RF 3.13. El usuario será capaz de visualizar y modificar si el servo funciona en modo digital o analógico.

RF 4. Pestaña Configuration.

RF 4.1. El usuario será capaz de visualizar el tipo del dispositivo conectado.

RF 4.2. El usuario será capaz de visualizar la versión software del dispositivo conectado.

RF 4.3. El usuario será capaz de visualizar el Cyclic Redundancy Code (CRC) del dispositivo conectado.

RF 4.4. El usuario será capaz de visualizar la versión del estimador del dispositivo conectado.

RF 4.5. El usuario será capaz de visualizar la versión software del POLAR (en caso de tenerlo) del dispositivo conectado.

RF 4.6. El usuario será capaz de activar/desactivar la opción de reporte de información del sensor de temperatura externa.

RF 4.7. El usuario será capaz de activar/desactivar la opción de reporte de información del sensor de altitud barométrica.

RF 4.8. El usuario será capaz de activar/desactivar el magnetómetro.

RF 4.9. El usuario será capaz de activar/desactivar la opción de reporte de información del sensor de consumo de los servos.

RF 4.10. El usuario será capaz de activar/desactivar la opción de reporte de información de los ángulos de los payloads conectados.

RF 4.11. El usuario será capaz de activar/desactivar la opción de establecer los ángulos fijos de los payloads conectados.

RF 4.12. El usuario será capaz de activar/desactivar la opción de plan de vuelo a cíclico.

RF 4.13. El usuario será capaz de activar/desactivar la opción de aceptar comandos para cambiar la velocidad aerodinámica.

RF 4.14. El usuario será capaz de activar/desactivar la alarma de comunicación fallida.

RF 4.15. El usuario será capaz de activar/desactivar la alarma de BINGO.

RF 4.16. El usuario será capaz de activar/desactivar la opción del cambio de la velocidad aerodinámica en mitad de un plan de vuelo.

RF 4.17. El usuario será capaz de activar/desactivar la opción del despliegue del paracaídas en caso de estar activado el modo seguro.

RF 4.18. El usuario será capaz de activar/desactivar la opción que permite que el UAV sea lanzado desde una catapulta.

RF 4.19. El usuario será capaz de activar/desactivar mejoras para el aterrizaje sobre el mar.

RF 4.20. El usuario será capaz de activar/desactivar la opción de aterrizaje de un avión de ala fija utilizando el paracaídas.

RF 4.21. El usuario será capaz de activar/desactivar la opción que permite el despliegue del paracaídas en caso de saltar la alarma de energía.

RF 4.22. El usuario será capaz de activar/desactivar la opción que permite el despliegue del paracaídas en caso de que la altitud esté por debajo de un mínimo.

RF 4.23. El usuario será capaz de activar/desactivar la opción que permite la liberación del paracaídas, ya sea iniciada por el usuario o automáticamente.

RF 4.24. El usuario podrá poner el POLAR en modo Programmable (PGM).

RF 4.25. La configuración actual del dispositivo podrá ser descargada.

RF 5. Pestaña Telemetry.

RF 5.1. El usuario será capaz de activar/desactivar el envío de paquetes adicionales de información.

RF 5.2. El usuario será capaz de activar/desactivar el envío de paquetes adicionales de información sobre el motor.

RF 5.3. El usuario será capaz de activar/desactivar la reducción del número de paquetes de telemetría.

RF 5.4. El usuario será capaz de activar/desactivar el envío de paquetes con información de depuración.

RF 5.5. El usuario será capaz de monitorizar las estadísticas de comunicación, incluyendo los bytes recibidos y enviados, los paquetes perdidos y enviados y la calidad del enlace de comunicación.

RF 5.6. El usuario será capaz de visualizar la ocupación del ancho de banda, tanto en bajada como en subida.

RF 5.7. El usuario será capaz de modificar el valor de timeout que determina que se ha perdido la señal con el dispositivo.

RF 5.8. La configuración actual del dispositivo podrá ser descargada.

RF 6. Pestaña Sensors.

RF 6.1. El usuario será capaz monitorizar los valores de los Analog-to-Digital Converter (ADC).

RF 6.2. El usuario será capaz monitorizar los valores de los sensores.

RF 7. Pestaña Ip Configuration.

RF 7.1. El usuario será capaz de visualizar y modificar la dirección IP del dispositivo conectado.

RF 7.2. El usuario será capaz de visualizar y modificar el puerto User Datagram Protocol (UDP) del dispositivo.

RF 7.3. El usuario será capaz de visualizar y modificar las direcciones destino a las que enviará el dispositivo.

RF 7.4. El usuario será capaz de activar/desactivar el modo broadcast.

RF 7.5. La configuración actual del dispositivo podrá ser descargada.

RF 8. Pestaña Gpio.

RF 8.1. Las opciones de los desplegables se corresponderán con las opciones del archivo "OWIO.xml" ubicado en la carpeta "Documents/Visionair/UserTools".

RF 8.2. El usuario será capaz de configurar los distintos pines GPIO como:

- Salidas: estas pueden configurarse como Pulse Width Modulation (PWM) (a velocidades de 50 Hz, 200 Hz y 400 Hz) o como discretas (a 0 o 3.3 voltios).
- Entradas: discretas o contador de pulsos.

RF 8.3. El usuario será capaz de establecer el número de canales usados (1-10) en el modo "Serial PWM" del GPIO 0.

RF 8.4. El usuario será capaz de establecer el ADC como un solo terminal o como diferencial.

RF 8.5. La configuración actual del dispositivo podrá ser descargada.

RF 8.6. La configuración actual de un dispositivo podrá ser almacenada en un fichero XML.

RF 8.7. La configuración de un dispositivo podrá ser cargada desde un fichero XML.

RF 8.8. La configuración actual de un dispositivo podrá ser descargada.

RF 9. Pestaña Communication Configuration.

Requisitos comunes.

RF 9.1. La configuración actual del dispositivo podrá ser descargada.

RF 9.2. La configuración actual de un dispositivo podrá ser almacenada en un fichero XML.

RF 9.3. La configuración de un dispositivo podrá ser cargada desde un fichero XML.

Requisitos para VECTOR.

RF 9.4. Las opciones de los desplegables se corresponderán con las opciones del archivo "CommConfigVector.xml" ubicado en la carpeta "Documents/Visionair/UserTools".

RF 9.5. El usuario podrá configurar el tipo de dispositivo conectado a los puertos serie cero, uno, dos, tres y el POLAR externo.

RF 9.6. El usuario podrá configurar la velocidad de transmisión de los puertos serie cero, uno, dos, tres y el POLAR externo.

RF 9.7. El usuario podrá configurar la paridad de la conexión de los puertos serie cero, uno, dos, tres y el POLAR externo.

RF 9.8. El usuario podrá configurar la polaridad de la conexión de los puertos serie cero, uno, dos, tres y el POLAR externo.

RF 9.9. El usuario podrá configurar el tipo de dispositivo conectado mediante los buses Controller Area Network (CAN)0 y CAN1.

RF 9.10. El usuario podrá configurar la velocidad de transmisión de los buses CAN0 y CAN1.

RF 9.11. El usuario podrá configurar los puertos COMmunications (COM) (del cero al tres) como puerto de payload, puerto de Logistics Information Network (LINK), IDLE, DATASOURCE o TUNNEL.

RF 9.12. El usuario podrá configurar el puerto Ethernet como puerto de LINK, IDLE o DATASOURCE.

RF 9.13. El usuario podrá establecer si se desean obtener los datos de un POLAR externo o del POLAR interno.

Requisitos para AP04.

RF 9.14. Las opciones de los desplegables se corresponderán con las opciones del archivo "CommConfigAp04.xml" ubicado en la carpeta "Documents/Visionair/UserTools".

RF 9.15. El usuario podrá configurar el tipo de dispositivo conectado a los puertos serie cero y uno.

RF 9.16. El usuario podrá configurar la velocidad de transmisión de los puertos serie cero y uno.

RF 9.17. El usuario podrá configurar la paridad de la conexión de los puertos serie cero y uno.

RF 9.18. El usuario podrá configurar la polaridad de la conexión de los puertos serie cero y uno.

RF 10. Pestaña Camera.

RF 10.1. Si la opción "Fixed payload" está activa, aparecerá un warning indicando que esta opción está activa y deshabilitará toda la interfaz.

RF 10.2. Si la opción "Report angles" no está activa, aparecerá un warning indicando que esta opción no está activa, y que por lo tanto, no se recibirá información sobre la cámara.

RF 10.3. El usuario será capaz de modificar el modo de la cámara.

RF 10.4. El usuario será capaz de modificar y monitorizar el giro de la cámara en el modo Aircraft.

RF 10.5. El usuario será capaz de modificar y monitorizar la inclinación de la cámara en el modo Aircraft.

RF 10.6. El usuario será capaz de modificar y monitorizar el rumbo de la cámara en el modo Ground.

RF 10.7. El usuario será capaz de modificar y monitorizar la elevación sobre el terreno en el modo Ground.

RF 10.8. El usuario será capaz de modificar y monitorizar la latitud de la cámara en el modo Geo.

RF 10.9. El usuario será capaz de modificar y monitorizar la longitud de la cámara en el modo Geo.

RF 10.10. El usuario será capaz de modificar y monitorizar la altitud de la cámara en el modo Geo.

RF 10.11. El usuario será capaz de modificar y monitorizar el incremento delta para el giro de la cámara.

RF 10.12. El usuario será capaz de modificar y monitorizar el incremento delta para la inclinación de la cámara.

RF 10.13. El usuario será capaz de modificar y monitorizar el valor del zoom de la cámara.

RF 10.14. El usuario será capaz de girar la imagen de la cámara en horizontal.

RF 10.15. El usuario será capaz de girar la imagen de la cámara en vertical.

RF 10.16. El usuario será capaz de resetear la configuración de la cámara.

RF 11. Pestaña Transponder.

RF 11.1. Las opciones de los desplegables se corresponderán con las opciones del archivo "XPDR.xml" ubicado en la carpeta "Documents/Visionair/UserTools".

RF 11.2. El usuario será capaz de modificar la dirección International Civil Aviation Organization (ICAO) del transponedor.

RF 11.3. El usuario será capaz de modificar el número de registro del transponedor.

RF 11.4. El usuario será capaz de modificar la velocidad máxima del aire del transponedor.

RF 11.5. El usuario será capaz de modificar la velocidad de transmisión del puerto COM 0.

RF 11.6. El usuario será capaz de modificar el origen de datos del GPS, esto es, si se deben obtener los datos del GPS integrado o de uno externo.

RF 11.7. El usuario será capaz de modificar el nivel de integridad del GPS.

RF 11.8. El usuario será capaz de modificar el tipo de UAV de categoría A.

RF 11.9. El usuario será capaz de modificar el tipo de UAV de categoría B.

RF 11.10. El usuario será capaz de modificar el valor de tamaño del UAV que almacena el transponedor.

RF 11.11. El usuario será capaz de modificar el offset utilizado para calcular la altitud del UAV.

RF 11.12. El usuario será capaz de modificar el Identification Data (ID) de vuelo.

RF 11.13. El usuario será capaz de modificar el código del transponedor.

RF 11.14. El usuario será capaz de modificar el origen del dato de altitud, esto es si utiliza el proporcionado por el transponedor o el proporcionado por el dispositivo.

RF 11.15. El usuario será capaz de modificar el modo del transponedor.

RF 11.16. El usuario será capaz de visualizar el estado del transponedor.

RF 11.17. El usuario será capaz de visualizar la altitud de origen del transponedor.

RF 11.18. El usuario será capaz de visualizar el modo del transponedor.

RF 11.19. El usuario será capaz de visualizar la altitud de presión del transponedor.

RF 11.20. El usuario será capaz de visualizar el tipo del transponedor.

RF 11.21. El usuario será capaz de visualizar el código del transponedor.

RF 11.22. El usuario podrá activar/desactivar la recepción de información periódica del transponedor.

3.2.2. Requisitos no funcionales

RNF 1. Lenguaje de programación

Como lenguaje de programación se utilizará C# sobre Visual Studio 2015 con el objetivo de construir todo el software empresarial sobre la misma plataforma, permitiendo una sencilla integración entre las diferentes aplicaciones y un fácil mantenimiento de toda la infraestructura software.

RNF 2. Estructura hardware

La aplicación podrá operar sobre sistemas hardware que cuenten con los mínimos requisitos que precisa la aplicación Visionair.

RNF 2.1. El sistema operativo puede ser uno de los siguientes: Microsoft Windows XP, Microsoft Windows 2000, Microsoft Windows Vista, Microsoft Windows 7 o Microsoft Windows 8.

RNF 2.2. Microprocesador: la aplicación corre tanto en sistemas de 32 bits como 64 bits.

RNF 2.3. RAM: mínimo 1 GB.

RNF 2.4. Video RAM: mínimo de 32 MB.

RNF 2.5. Disco duro: mínimo 32 MB.

RNF 2.6. Es necesario tener instalados los siguientes paquetes: Microsoft .NET Framework 4 o superior, Microsoft Windows Installer 3.0 o superior, Microsoft Visual C++ 2005 SP1, Microsoft Visual C++ 2010.

RNF 3. Idioma

La aplicación está escrita en inglés utilizando las abreviaturas aceptadas por la empresa y no se permite el cambio a otro idioma de forma manual.

RNF 4. Diseño de la interfaz

RNF 4.1. El panel tendrá el nombre de "User Tools".

RNF 4.2. Las diferentes acciones de configuración serán agrupadas en pestañas atendiendo al criterio elegido por los ingenieros del grupo Flight Control.

RNF 4.3. Todos los títulos y encabezados de cada sección tendrán el mismo formato, texto en negrita y primera letra de cada palabra en mayúscula.

RNF 4.4. Todos los elementos de la interfaz mostrarán un texto de ayuda al segundo de haber situado el ratón sobre ellos.

RNF 4.5. Las opciones de los menús desplegables estarán ordenadas alfabéticamente, los primeros elementos de la lista corresponderán a las opciones que empiezan por A y los últimos a las opciones que empiezan por Z.

RNF 4.6. Los botones tornarán en rojo cuando la acción que se ha ordenado no se haya ejecutado correctamente.

RNF 5. Instalación

El software se diseñará como una librería estándar (DLL) cuya instalación consistirá en la copia del archivo en la carpeta específica de Visionair, "Extensions".

RNF 6. Tiempo de respuesta

Los mensajes serán enviados repetidas veces a la espera de una respuesta. Se repetirá la acción hasta un máximo de 8 veces cada 12 milisegundos.

RNF 7. Evolutivo

La aplicación deberá soportar modificaciones e implementaciones futuras atendiendo a nuevos requisitos.

RNF 8. Revisiones periódicas

Con el fin de garantizar el buen funcionamiento de la aplicación, se realizarán revisiones una vez al mes con el objetivo de corregir errores y mejorar el software. Estas revisiones incluyen:

- Revisión de la librería de comunicaciones: se modificarán mensajes cuya estructura haya variado y se añadirán nuevos, en caso de haberlos.
- Limpieza de funciones obsoletas.
- Adición de nuevas funcionalidades.
- Realización de ATP para comprobar que todo funciona correctamente.
- Documentación de los cambios realizados.
- Aprobación por el supervisor para su incorporación al Knowledge Base.

RNF 9. Concurrencia/Transacciones

El intercambio de información entre el dispositivo y la aplicación se realizará a través del driver de comunicaciones de Visionair, el cual garantizará que el mensaje sea empaquetado de acuerdo a la estructura de paquetes que admite el dispositivo. Esta comunicación con el dispositivo se realizará a través de puerto COM, conexión UDP o Transmission Control Protocol (TCP).

RNF 10. Disponibilidad

La aplicación deberá estar disponible para los usuarios en todo momento, las 24 horas del día, los 7 días de la semana. Para tal propósito, se mantendrá siempre una versión estable en el Knowledge Base.

RNF 11. Usabilidad

- El sistema debe proporcionar mensajes de error que sean informativos.
- El sistema debe contar con un módulo de ayuda en línea.
- Todas las funciones serán únicas, es decir, no estarán repetidas.

Capítulo 4

Diseño

En esta sección, se describe detalladamente el diseño del sistema creado, que pasa por un análisis de la estructura general y termina con un análisis de los subsistemas que componen la estructura del proyecto.

4.1. Diseño general

En la Figura 4.1 se muestra el esquema general del sistema diseñado. El sistema se compone de tres partes:

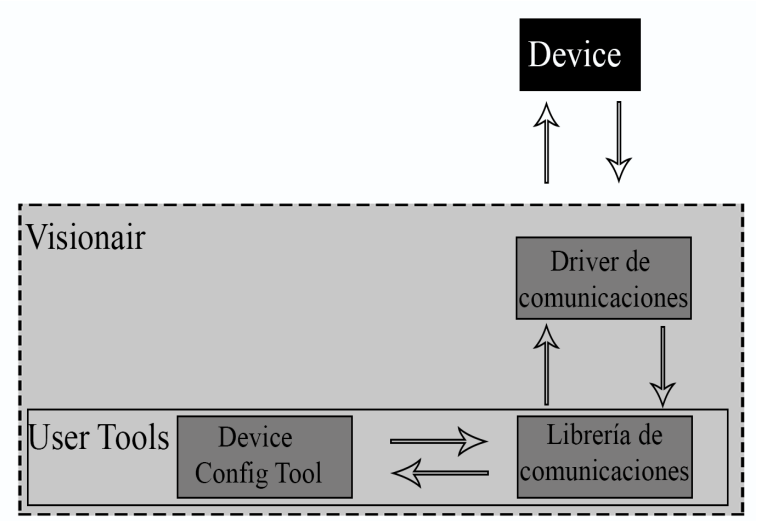


Figura 4.1: Esquema general del sistema.

1. **El Device [10]**: el dispositivo puede ser un VECTOR o un AP04. Este dispositivo está transmitiendo y recibiendo datos continuamente de Visionair mediante una conexión por Ethernet o por puerto serie.

2. **Visionair [11]**: la estación de tierra Visionair posee muchas funciones, de las cuales van a ser utilizadas el driver de comunicaciones y la capacidad de cargar DLL. La librería de comunicaciones de Visionair no es más que un driver que se encarga de abrir un puerto serie o una conexión Ethernet, leer todos los datos que llegan por ese puerto, descartar todos aquellos que no son correctos e informar a todos aquellos subsistemas subscritos a sus eventos de la llegada de nuevos datos. Visionair clasifica los datos recibidos del dispositivo en varias familias de paquetes, atendiendo al valor de su primer identificador y, en función de este, se lanzan diferentes eventos para cada familia. Por otro lado, este driver de comunicaciones también permite a los subsistemas enviar datos por el puerto abierto al dispositivo, siempre y cuando el formato de estos datos sea un formato específico (definido en el ICD).

3. **User Tools**: las User Tools son un conjunto de cuatro aplicaciones software que dotan al usuario poseedor de un hardware de UAV Navigation de un conjunto de herramientas de configuración. Estas cuatro aplicaciones son "Software Update Tool", utilizada para cargar software en el dispositivo, "Log File Converter", la cual convierte los datos de un plan de vuelo de un formato de datos a otro, "GCS Config Tool", que proporciona herramientas para la configuración de la radio y "Device Config Tool" de la que es objeto este proyecto. "Device Config Tool" es una aplicación que proporciona herramientas para la configuración de diferentes parámetros del dispositivo como pueden ser el movimiento de los servos, la gestión de los diferentes puertos de comunicación del dispositivo, la configuración de ganancias o, incluso, la calibración de diferentes cargas útiles del dispositivo como pueden ser el transponedor o una cámara. Todo esto lo hace a través de una serie de mensajes generados y descompuestos por la librería UAVNCOMM de las User Tools.

4.2. Diseño de la librería de comunicaciones

La librería de comunicaciones, como se ha dicho anteriormente, se trata de un módulo del programa que se nutre de los datos que Visionair obtiene del dispositivo conectado para su uso particular. Se destacan tres partes, que son las que se muestran en la Figura 4.2.

Por una parte, se encuentra el paquete de los identificadores. Se trata de un paquete formado solo por enumerados, los identificadores de las familias de paquetes, que son los que se recogen en el paquete ID, y los enumerados de parámetros concretos, como pueden ser los modos de una cámara, las diferentes paridades que se pueden configurar en los puertos de comunicación o los diferentes tipos de sensores.

Después, se encuentra el paquete EventArguments. Este paquete está formado por clases que definen los argumentos de cada uno de mensajes que puede enviar el dispositivo. Todas estas clases implementan la clase EventArgs [12], ya que serán enviadas como parámetros de eventos. Por ejemplo, el paquete "GetPortCfg" contiene la configuración de un determinado puerto de comunicación, el identificador del puerto al cual pertenece la información, la velocidad de transmisión, la paridad y la polaridad de la conexión. De esta forma, la clase RxPortConfigEventArgs tendrá la forma que se muestra en la Figura B.1.

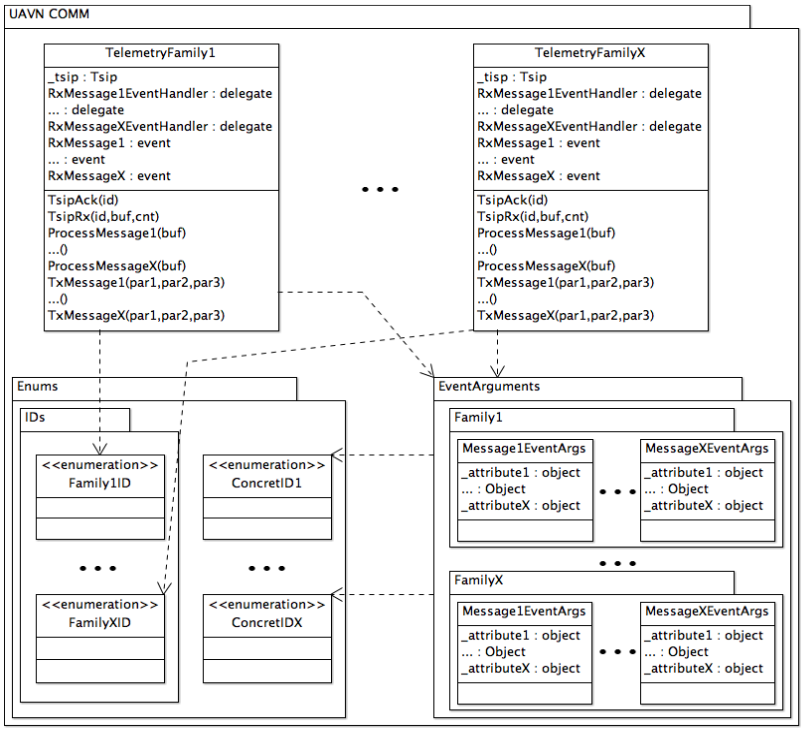


Figura 4.2: Esquema de la librería de comunicaciones.

Por último, se encuentra el núcleo de la librería, el cual está formado por diversas clases que representan a las diferentes familias de paquetes. Estas clases están subscritas a los eventos que lanza la librería de comunicaciones de Visionair, de manera que, por ejemplo, cuando un paquete de la Familia 1 es enviado por el dispositivo, Visionair lanza un evento que informa del suceso, el cuál es capturado por la clase que representa a la Familia 1. Una vez se ha capturado el evento, la función TsipRx o TsipAck, dependiendo de si el mensaje recibido contiene información o un Acknoleadge (ACK), identifica qué mensaje es para después llamar a la función que se encarga del desglose de ese paquete en concreto. Una vez se ha llamado a esta función, se identifican todos los parámetros que contiene el mensaje y se crea una clase como la de la Figura B.1. Posteriormente se lanza un evento donde esta clase es enviada como parámetro.

4.3. Diseño de la aplicación

Por último, se va a explicar la arquitectura de la aplicación "Device Config Tool". Esta se compone de una parte gráfica y otra que contiene toda la lógica. Debido a esta composición, se ha decidido realizar una implementación que aplique el patrón de diseño Model View ViewModel (MVVM) [13], cuya estructura es la que se muestra en la Figura 4.3.

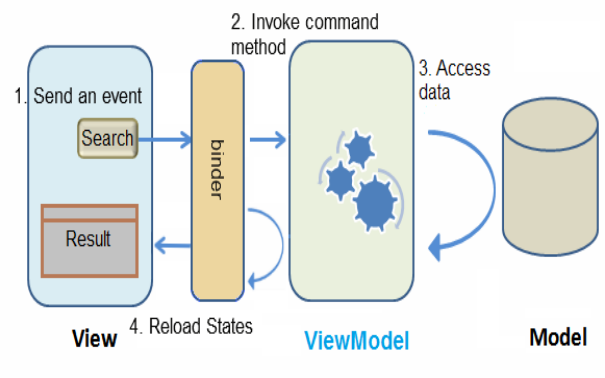


Figura 4.3: Modelo MVVM [14].

Este modelo posee una vista cuyos elementos gráficos tienen sus valores ligados a variables definidas en el ViewModel, que a su vez están ligadas a los datos del modelo. De esta forma, cuando el usuario modifica el valor de algún elemento de la interfaz, éste se modifica automáticamente en el ViewModel y a su vez en el modelo, desencadenando, o no, una serie de acciones que dependen de la lógica interna definida. Y al contrario, si alguna de estas variables es modificada en el ViewModel, automáticamente es actualizada tanto en el modelo como en la interfaz sin necesidad de hacerlo manualmente. Gracias a esto, un mismo ViewModel puede utilizarse con diversas vistas. Este aspecto es muy importante para este proyecto, ya que esta aplicación tiene como objetivo futuro su implementación para desarrolladores. Este modelo, por lo tanto, proporciona cierta flexibilidad a la hora de ampliar la funcionalidad, corregir errores de comportamiento o, incluso, actualizar funciones porque el tratamiento de sus datos haya variado.

Por otro lado, se han definido una serie de clases que proveen de utilidades al ViewModel, mostradas en la Figura A.1. De estas cuatro clases, se ha decidido que tres de ellas sean estáticas, ya que no dependen de ningún objeto para su funcionamiento.

Luego, el diseño final del proyecto sería el de la Figura 4.4. En el Capítulo 5, se explicará más en profundidad cómo están implementados cada uno de estos paquetes de clases.

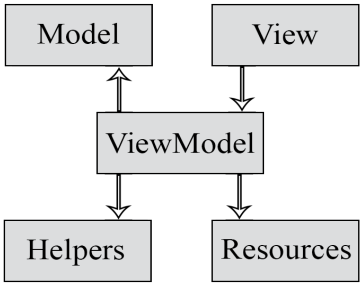


Figura 4.4: Esquema general de la aplicación.

Capítulo 5

Desarrollo

En este capítulo se va a describir de forma detallada la implementación del sistema desarrollado. Antes de comenzar con el análisis de la estructura del código, se va a proceder a exponer las características del hardware y el software utilizados en el proyecto, así como las plataformas empleadas en su desarrollo.

5.1. Equipo de desarrollo

5.1.1. Hardware

Las características hardware del equipo de desarrollo debían cumplir los requisitos mínimos que Visionair necesita para funcionar. Así pues, el proyecto se ha desarrollado en un MacBook Pro con procesador Intel Core i5 de 2,4 GHz y 8GB de RAM con sistema operativo OS X El Capitan, en el que se ha montado una máquina virtual que ha sido provista de 3GB de memoria RAM, dos núcleos del procesador, 60GB de disco duro SSD, y en la que se ha instalado un sistema operativo Windows 7 Ultimate.

Por otro lado, también han sido necesarios dos pilotos automáticos de la empresa UAV Navigation para comprobar el funcionamiento de la aplicación. Uno de estos pilotos automáticos es un dispositivo AP04, y el otro es un dispositivo VECTOR, ambos provistos de dos CPU.

Por último, ha sido necesario un cable RS-232 para poder comunicar Visionair con los dispositivos y un cable J-Link para poder depurar el código interno de los autopilotos.

5.1.2. Software

En cuanto a las características del software, hay que diferenciar entre el software utilizado en los autopilotos y el utilizado en el ordenador.

Software utilizado en el ordenador

En el ordenador, ha sido necesaria, tanto para el desarrollo de la librería de comunicaciones como para el desarrollo de la aplicación, la instalación del entorno de desarrollo Visual Studio 2015 Community.

Además, también ha sido necesaria la instalación de una distribución de Látex [15] para la redacción de la presente memoria.

Software utilizado en los autopilotos

Como se ha comentado en el Apartado 5.1.1, los autopilotos poseen dos CPU. En estas CPU se han cargado dos tipos de software, uno en cada una de ellas. En la CPU0 se ha cargado un software de avión y en la CPU1 se ha cargado un software de helicóptero.

5.2. Plataformas

5.2.1. Visual Studio 2015 Community

Visual Studio 2015 Community es un IDE integrado que permite la creación de aplicaciones para Windows, Android e iOS, aplicaciones web y servicios en la nube. Permite desarrollar aplicaciones en diversos lenguajes, como son C#, Visual Basic, F#, C++, Python, Node.js y HTML/JavaScript [16].

Este ha sido el entorno de trabajo escogido por el jefe del departamento de informática de UAV Navigation para el desarrollo del presente proyecto. Esta decisión se ha basado en el hecho de que, por un lado, se deseaba utilizar WPF para el desarrollo de la interfaz y, por otro lado, se quería que el programa desarrollado se integrase fácilmente con Visionair.

5.2.2. C#

C# es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET. C# utiliza un modelo de objetos similar al de java y su sintaxis es sencilla, ya que deriva de la de C/C++ [17].

C# ha sido el lenguaje empleado para el desarrollo del proyecto, y no cualquiera de los soportados por Visual Studio, principalmente porque la empresa pretende migrar todo el código que posee a este lenguaje, ya que es fresco, actual y orientado a objetos.

5.2.3. WPF

WPF es una tecnología de Microsoft que permite el desarrollo de interfaces de interacción en Windows que posee características de aplicaciones Windows y web [18].

Actualmente, Visionair posee un mecanismo que le permite cargar DLL cuya interfaz de programación sea Windows Form (WF). Debido a que WF se ha quedado un poco obsoleto desde la aparición de WPF, se ha decidido utilizar un mecanismo que permitiese cargar una aplicación desarrollada con WPF en un WF para evitar tener que modificar este mecanismo de carga.

Gracias a esto, finalmente se ha podido utilizar WPF para la implementación de las vistas de la aplicación.

5.2.4. Adobe Photoshop CC

Adobe Photoshop es un editor de gráficos rasterizados, desarrollado por Adobe Systems Incorporated, que se utiliza para el retoque de imágenes [19].

Esta herramienta de edición gráfica ha sido utilizada para la creación de las imágenes de la aplicación por ofrecer una infinidad de herramientas que facilitaban el diseño de estas.

5.2.5. Visionair

Visionair es un software estándar de la estación de tierra perteneciente a la empresa UAV Navigation y utilizado para la planificación y ejecución de misiones con UAV.

Este programa ha sido utilizado como intermediario entre la aplicación y el dispositivo. En el arranque, Visionair carga la DLL que contiene la librería de comunicaciones y la aplicación, y muestra la interfaz de esta. Gracias a este software, se ha podido verificar el correcto funcionamiento del proyecto.

5.2.6. GanttProject

GanttProject es una aplicación utilizada para la programación y gestión de proyectos [20] que permite elaborar diagramas temporales que recogen la duración de las tareas presentes en un proyecto.

La aplicación ha sido utilizada para la confección de las etapas del proyecto.

5.2.7. Bitbucket

Bitbucket es una página que ofrece un servicio de alojamiento para los proyectos que utilizan un sistema de control de versiones [21].

Bitbucket se ha utilizado par mantener el control de las versiones, facilitando, sobre todo, la corrección de errores. Otra de las razones del uso de este servicio ha sido para que el tutor del proyecto pudiese ver el progreso e indicar fallos, posibles mejoras u otro tipo de comentarios.

5.2.8. Sublime Text

Sublime Text es un editor de texto y de código fuente [22].

Esta herramienta ha sido utilizada para la redacción de la presente memoria y para la edición de algunas clases del proyecto.

5.3. Estructura del código

A continuación se va a detallar la implementación del código, pasando por el conjunto de recursos, la librería de comunicaciones y por último la aplicación.

5.3.1. Recursos

Imágenes

En esta carpeta se recogen las imágenes utilizadas en la interfaz. Estas imágenes están en formato "png" y han sido diseñadas con la herramienta Photoshop. Entre estas imágenes se recogen las que componen la brújula de la pestaña Basic y las que representan los valores Pan y Tilt de la pestaña Camera.

Estilos

Esta carpeta contiene un archivo de recursos en el que se han definido todos los estilos gráficos de la aplicación como pueden ser los márgenes, estilo de las etiquetas, las cajas de texto o los botones.

La creación de este archivo nace de la idea de mantener una interfaz estándar y minimizar al máximo el trabajo a realizar cuando se desea modificar algún aspecto visual.

Textos

La aplicación desarrollada está escrita en inglés, sin embargo, se desea que, en un futuro, esta pueda ser traducida a otros idiomas. Por este motivo, se han definido un conjunto de archivos que recogen los textos que se muestran en la interfaz de la aplicación en lugar de definir estas cadenas en los propios archivos de las vistas.

XML

Se desea que la aplicación cargue de forma dinámica las opciones de los menús desplegables, por ello, se han definido unos archivos XML que almacenan las opciones por defecto de estos desplegables. Al iniciar la aplicación, se crea una carpeta donde se copian estos archivos para que los usuarios pueden modificarlos y definir sus propias opciones. Si en un momento dado se desean recuperar los XML originales, bastaría con eliminar la carpeta que la aplicación crea con estos archivos y arrancar de nuevo el programa.

Utilidades

Las utilidades están formadas por un conjunto de cuatro clases que recogen una serie de funciones auxiliares utilizadas tanto en la librería de comunicaciones como en la aplicación.

A continuación se va a explicar cada una de las clases.

- **DataConverter**: es una clase estática cuyos métodos tienen la función de obtener el valor numérico a partir de un string, el cual es pasado mediante parámetro. Si en la extracción del valor numérico del string ocurriese cualquier error, es decir, no hubiese un número en el string, el número se encontrase acompañado de letras o cualquier caso similar, estas funciones devuelven excepciones que son tratadas en cada una de las clases que utilizan estas funciones.

- **InterfaceUtils**: esta clase, al igual que la anterior, es una clase estática. La clase posee dos métodos cuyo cometido es el de abrir un cuadro de diálogo para guardar y abrir archivos. Las ventanas de diálogo, únicamente, permiten abrir y guardar archivos XML.

  Esta clase es utilizada por aquellas pestañas de configuración que permiten guardar y cargar una configuración almacenada en un XML, como puede ser la pestaña de Servos o la pestaña GPIO.

- **Utils**: este proyecto se basa en la recepción y la transmisión de datos. Estos datos se encapsulan en mensajes y, tanto para su encapsulación dentro del mensaje como para la obtención de su valor, son necesarias un conjunto de funciones que ayuden con esta tarea. De esta necesidad surge la clase Utils.

La clase estática Utils contiene métodos que permiten, pasado un array de valores, obtener un valor que se encuentra en una determinada posición o, por el contrario, encapsular un cierto valor dentro de un array de datos.

Por otro lado, esta clase también contiene funciones que permiten manipular diccionarios. Estas funciones son dos, una que permite crear diccionarios a partir de datos contenidos un XML y otra que ordena los datos de este diccionario por orden alfabético.

- **XmlUtils**: por último, se encuentra la clase XmlUtils. Esta clase contiene utilidades necesarias para la lectura y la escritura de archivos XML.

5.3.2. Librería de comunicaciones

Como se ha explicado en el Capítulo 4, la librería de comunicaciones consta de un conjunto de identificadores, un conjunto de clases que representan cada mensaje junto con sus atributos y varias clases principales, una por cada familia de mensajes, que son las encargadas de capturar los eventos de los mensajes, decodificarlos y también es la responsable de la confección y el envío de mensajes.

Antes de explicar el funcionamiento de estas clases, es necesario aclarar que dentro de cada familia de mensajes hay dos tipos, los mensajes que contienen parámetros de configuración del autopiloto y los mensajes ACK, que son mensajes enviados por el autopiloto para informar a su emisor de que se ha recibido y procesado el mensaje de configuración correctamente.

Tras esta pequeña aclaración, se procede a explicar el funcionamiento de esta librería.

Las clases principales, como se ha comentado anteriormente, son las encargadas del tratamiento de mensajes. Cada una de estas clases tiene una instancia de la clase Tsip, clase principal de la librería de comunicaciones de Visionair, y están subscritas a los eventos que esta clase lanza por cada mensaje que llega. De esta manera, la clase TelemetryFamily1 está subscrita al evento TsipRxFamily1 y TsipAckFamily1. Además, estas clases tienen definidos una serie de eventos, uno por cada mensaje, los cuales son lanzados cuando se captura un evento de ACK o tras descomponer un mensaje. A estos eventos están subscritos los diferentes ViewModel, más adelante se explicará por qué.

Por otro lado, se distinguen otros dos tipos de funciones dentro de estas clases, las funciones de descomposición y las funciones de creación de mensajes. Cuando se recibe un mensaje, Visionair lo clasifica dentro de un determinado tipo de familia y lanza un evento para informar a las clases subscritas de que un mensaje de ese tipo ha llegado. Ese evento es capturado por las clases principales de las que se hablaba anteriormente, las cuales determinan el mensaje que es y proceden a llamar a una función de descomposición que desglosa el mensaje y extrae de él cada uno de los campos que contiene. Estas funciones de descomposición reciben dos argumentos, el búfer que contiene el mensaje y el tamaño del mismo. Tras desglosar el mensaje se crea una clase del tipo de mensaje recibido y se lanza un evento que posee como argumento la nueva clase creada (Figura 5.1).

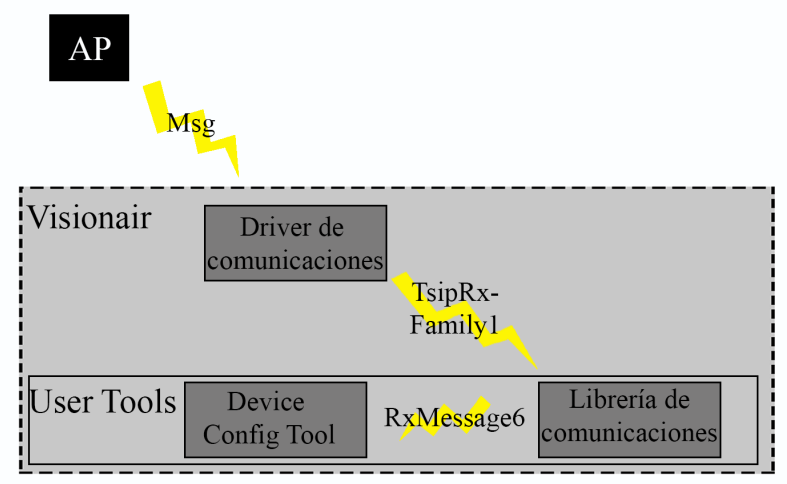


Figura 5.1: Proceso de recepción y descomposición de un mensaje desde el autopiloto.

El otro tipo de funciones tienen la misión de crear mensajes específicos y enviarlos utilizando las funciones de la clase Tsip. Dependiendo del mensaje que se desee crear, estas funciones reciben un número diferente de parámetros, obtenidos de la interfaz, los cuales son encapsulados en un array para su posterior envío.

5.3.3. Device Config Tool

Device Config Tool es la herramienta principal del proyecto. La herramienta está constituida por los paquetes que a continuación se describen.

Helpers

El paquete Helpers contiene una serie de clases que sirven de apoyo a los ViewModel.

Por un lado están las interfaces (Figura C.1):

- **ISectionAction**: es una interfaz que contiene declaraciones de funciones, propiedades y eventos que una clase debe implementar para la utilización de la vista ControlButtonsView. Se definen funciones para los botones "Download", "Upload", "Open" y "Save" y eventos para controlar el estado de la barra de progreso, la finalización de la acción del botón o la aparición de un error.

- **ILoopInterface**: algunas pestañas precisan de un mecanismo que active un bucle dedicado al envío periódico de mensajes. Es por esto que se decidió crear la interfaz ILoopInterface, la cual contiene dos declaraciones de funciones, InitLoop y StopLoop. Actualmente, la única clase que implementa esta interfaz es CameraViewModel.

Por otro lado se encuentran las clases de soporte (Figura C.2):

- **DelegateCommand**: clase que implementa la interfaz ICommand, creada para asignar acciones a los botones de las interfaces.

- **ExpressionExtensions**: es una clase que permite escribir OnPropertyChanged(() =>PropertyName) en lugar de OnPropertyChanged("PropertyName"). La ventaja de esto es que el código es mucho más fácil de mantener frente a modificaciones en los nombres de las propiedades. Si se modifica el nombre de una propiedad y se compila el código, el compilador avisaría de un error en una propiedad que no existe.

- **ObservableExtension**: clase estática que contiene una serie de métodos cuya finalidad es la de realizar una acción en un hilo de ejecución diferente al principal. Existen dos funciones principales, WaitforResponse y ToUAVObservableSequentialPattern. La primera realiza una acción repetidas veces, cada cierto periodo de tiempo establecido previamente, hasta que ocurre una acción. En este caso, se envía un mensaje repetidas veces hasta obtener el ACK del dispositivo. Además, contiene mecanismos que informan del progreso de la acción, la aparición de errores o de la finalidad de la acción. La segunda función es igual que la primera, salvo que esta realiza las acciones en bucle. Por ejemplo, cuando se quieren calibrar los servos, se utiliza esta última función, la cual envía en bucle el mensaje de configuración para cada uno de ellos.

- **ViewModeBase**: clase que implementa la interfaz INotifyPropertyChanged [23]. Define un método que es utilizado por los ViewModels para informar a la interfaz de que hay cambios en los datos del modelo.

Modelos

Los modelos son un conjunto de clases que se encargan de contener los datos que representan a cada una de las pestañas de la aplicación. No es necesario explicar mucho más sobre estas clases, ya que únicamente tienen atributos y propiedades. La Figura C.3 muestra el diagrama de clases de los modelos.

Vistas

Las vistas definen la interfaz de la aplicación y muestran los datos del modelo. Estas están diseñadas mediante WPF y definen sus estilos y textos a través del diccionario de recursos (Apartado 5.3.1) y los archivos de recursos de textos (Apartado 5.3.1) respectivamente.

Los archivos de vistas que posee la aplicación son los que se listan:

- **MainWindow**: vista que carga todas las pestañas de la aplicación. Debido a que la aplicación da servicio a dos tipos de dispositivos, algunas de las pestañas son diferentes para cada uno de ellos. Así pues, el MainWindow cargará unas vistas u otras atendiendo al tipo de dispositivo conectado.
- **ControlButtonsView**: parte de la vista que es común a varias pestañas, posee los botones "Download", "Upload", "Open" y "Save".
- **AircraftView**: proporciona las herramientas necesarias para la modificación de las ganancias del dispositivo (Figura F.1).
- **BasicView**: esta pestaña muestra datos relacionados con el dispositivo (Figura F.2).
- **CameraView**: proporciona las herramientas necesarias para la configuración de una cámara externa (Figura F.7).
- **CommunicationConfigurationVectorView**: proporciona las herramientas necesarias para la configuración de los puertos de comunicación de un VECTOR (Figura F.4).
- **CommunicationConfigurationAp04View**: proporciona las herramientas necesarias para la configuración de los puertos de comunicación de un AP04 (Figura F.3).
- **CommunicationConfigurationView**: carga una de las dos vistas anteriores en función del dispositivo conectado.
- **ConfigurationView**: proporciona las herramientas necesarias para la configuración de las opciones de vuelo, los sensores y los payloads (Figura F.5).
- **IpConfigView**: proporciona las herramientas necesarias para la configuración de las direcciones IP de comunicación (Figura F.8).
- **OwioView**: proporciona las herramientas necesarias para la calibración de los pines GPIO (Figura F.6).
- **PayloadsView**: contenedor que muestra las pestañas de los payloads disponibles para su configuración.
- **SensorsView**: permite monitorizar datos de los sensores (Figura F.11).
- **ServosVectorView**: proporciona las herramientas necesarias para la calibración de los servos en un VECTOR (Figura F.10).
- **ServosAP04View**: proporciona las herramientas necesarias para la calibración de los servos en un AP04 (Figura F.9).
- **ServosView**: contiene la parte común de las dos vistas anteriores y, además, carga una u otra en función del dispositivo conectado.
- **TelemetryView**: muestra estadísticas de comunicación y permite configurar opciones de la telemetría (Figura F.12).

- **XPDRView**: proporciona las herramientas necesarias para la configuración de un transponedor (Figura F.13).

ViewModels

Los ViewModels (Figura C.4) son un conjunto de clases que contienen la lógica de la aplicación, ejecutan acciones en los datos del modelo para mostrarlos en la vista e interpretan los datos de la vista para modificarlos en el modelo.

En general, todos los ViewModels implementan o utilizan:

- **La clase ViewModeBase**: de esta forma pueden informar a la interfaz de que hay cambios en los datos del modelo.
- **La clase ObservableExtension**: utilizada para el envío de mensajes.
- **La interfaz ISectionAction**: es implementada por aquellos que precisan definir acciones para los botones de la vista ControlButtonsView.
- **La interfaz DelegateCommand**: implementada por aquellos que tienen que definir acciones para los botones de su vista.

Existen casos especiales cuya implementación es algo más compleja por dos motivos:

- **La vista y el modelo es el mismo, pero no el procedimiento para configurar los parámetros**. Este es el caso de AircraftViewModel e IpConfigViewModel. Para esta implementación, existen tres ViewModel, uno para cada dispositivo y uno que contiene las partes comunes, como son el modelo o las funciones para cargar y guardar una configuración. Las clases de los dispositivos implementan la interfaz ISectionAction, por lo que, cuando se crea el ViewModel común, este no sabe qué se le está pasando por parámetro, si una clase de VECTOR o una de AP04. La Figura D.1 muestra la implementación del ViewModel para la pestaña Aircraft. Como se puede observar, en AircraftViewModel existe una instancia de una clase del tipo ISectionAction, que puede ser la específica de VECTOR o AP04, y se relaciona con ella mediante las funciones que se establecen en la interfaz, "OnDownload", "OnUpload", "Save" y "Open" y mediante los eventos "ErrorInAction", "FinishAction" y "UpdateProgress".
- **La vista y el modelo son diferentes, pero poseen partes comunes**. Es el caso de ServosViewModel y CommunicationConfigurationViewModel. El conflicto se resuelve de forma similar a los anteriores. Se crean tres clases, cada una con su vista y modelo propio. Las dos clases específicas, para VECTOR y AP04, implementan la interfaz ISectionAction. En la Figura D.2 puede apreciarse que la clase ServosViewModel contiene el modelo común a los dos dispositivos, "_servos", y, como en el caso anterior, una instancia de una clase del tipo ISectionAction. Por otro lado ServosAp04ViewModel tiene su propio modelo, "_servosAp04" y ServosVectorViewModel el suyo, "_servosVector".

Capítulo 6

Pruebas y resultados

A continuación, se muestran las pruebas a las que ha sido sometida la aplicación y la librería de comunicaciones, entre las que se encuentran pruebas unitarias, pruebas de sistema y de integración y pruebas de validación.

6.1. Pruebas unitarias

En la fase de pruebas unitarias se ha comprobado el correcto funcionamiento de cada uno de los módulos de la librería de comunicaciones y la aplicación.

Las pruebas unitarias de la librería de comunicaciones han consistido en la verificación de la correcta composición y descomposición de los paquetes enviados y recibidos a través del driver de comunicaciones, es decir, si los paquetes cumplían con lo establecido en el ICD.

En el caso de la aplicación, se ha comprobado que la estructura MVVM se había aplicado de forma correcta, que la transición de los datos entre las distintas clases era la que se esperaba, que se habían definido todas las funciones necesarias para el tratamiento de la información de la interfaz y que estas se ejecutaban correctamente.

6.2. Pruebas de sistema y de integración

En las pruebas de sistema y de integración se han definido dos fases.

En la primera se ha comprobado que la librería de comunicaciones y la aplicación eran capaces de intercambiar la información de comunicación sin problema, de manera que la aplicación utilizase correctamente las funciones y los eventos de la librería de comunicaciones.

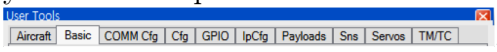
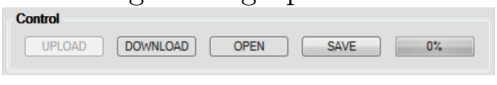
En la segunda fase se ha comprobado que, tanto la librería de comunicaciones como la aplicación, conseguían integrarse correctamente con Visionair, es decir, que la librería de comunicaciones era capaz de utilizar de forma correcta el driver de comunicaciones de Visionair y que la aplicación podía cargarse sin ningún problema dentro de este software de la estación de tierra.

En ambas fases se ha conseguido con éxito la integración de las partes.

6.3. Pruebas de validación

En las pruebas de validación se ha comprobado que cada uno de los elementos de la interfaz ejecutaba las acciones que se esperaba de ellos. A continuación, se muestran las pruebas generales a las que ha sido sometida la aplicación y las pruebas concretas de la pestaña Basic, el resto de pruebas pueden verse en el Anexo E.

6.3.1. General

Nº	Descripción	Resultado esperado	Test
1	Se conecta un AP04 o un VECTOR.	Aparecen las siguientes pestañas en el mismo orden, los mismos nombres y la misma pestaña seleccionada. 	OK
2	No se conecta ningún dispositivo. Se ejecuta la herramienta Device Config Tool.	Aparece el siguiente texto: “Connect a device to load User Tools”.	OK
3	Se abre la carpeta “Documents/Visionair” y se borra la carpeta “User Tools”. Se ejecuta la herramienta Device Config Tool.	La carpeta “User Tools” es creada de nuevo y contiene los XML por defecto.	OK
4	Se verifica que en cada pestaña aparece el siguiente grupo de botones: 	El botón “Upload” aparece deshabilitado, se debe descargar la configuración antes de poder actualizarla.	OK
5	Se presiona el botón “Upload”, “Download”, “Open” o “Save” en cualquier pestaña.	El grupo de botones “Control” aparece deshabilitado hasta que termina la acción.	OK
6	Se presiona el botón “Download” en cualquier pestaña.	Si hay algún dispositivo conectado, la configuración se descargará correctamente y todos los botones se habilitarán.	OK

Nº	Descripción	Resultado esperado	Test
7	Se presiona el botón “Download” en cualquier pestaña.	Si hay algún dispositivo conectado y la configuración no se descarga correctamente, todos los botones se habilitarán y el botón “Download” se pondrá de color rojo.	OK
8	Se presiona el botón “Download”, en cualquier pestaña.	Si no hay un dispositivo conectado, todos los botones se habilitarán y el botón “Download” se pondrá de color rojo.	OK
9	Se define una configuración en cualquier pestaña. Se presiona el botón “Upload”. Se presiona el botón “Download”.	Si hay algún dispositivo conectado y no ha ocurrido ningún error, la configuración será actualizada en el dispositivo y todos los botones se habilitarán.	OK
10	Se define una configuración en cualquier pestaña. Se presiona el botón “Upload”. Se presiona el botón “Download”.	Si hay algún dispositivo conectado y ocurre algún error, la configuración no será actualizada en el dispositivo, todos los botones se habilitarán, el botón “Upload” se pondrá de color rojo y la interfaz mostrará la configuración que el dispositivo posee en ese momento.	OK
11	Se define una configuración en cualquier pestaña. Se presiona el botón “Upload”.	Si no hay ningún dispositivo conectado el botón “Upload” se pondrá de color rojo.	OK
12	Se presiona el botón “Open” y se selecciona un archivo.	Si el archivo es un archivo de configuración correcto, los valores se cargarán.	OK
13	Se presiona el botón “Open” y se selecciona un archivo.	Si el archivo no es un archivo de configuración correcto, los valores no serán cargados y el botón “Open” se pondrá de color rojo.	OK
14	Se presiona el botón “Save” y se selecciona una carpeta donde guardar el archivo XML de configuración.	Se ha creado un archivo XML de configuración y contiene la configuración actual.	OK
15	Se conecta un AP04 o un VECTOR. Se ejecuta la aplicación. Se desconecta el dispositivo. Se pulsa cualquier botón.	El botón que se haya pulsado se pondrá de color rojo, exceptuando los botones “Open” y “Save”.	OK
16	Se abre cualquier pestaña. Se sitúa el ratón durante un segundo sobre cualquier botón, checkbox o caja de texto.	Aparece un Tooltip con la descripción del elemento.	OK

Tabla 6.1: Pruebas generales.

6.3.2. Pestaña Basic

Nº	Descripción	Resultado esperado	Test
1	Se abre la pestaña “Basic”.	En la parte superior derecha de la ventana aparece una brújula que muestra el valor de Yaw. El valor de Yaw mostrado solo toma valores entre 0 y 360. El valor de Yaw mostrado cambia cuando se rota el dispositivo.	OK
2	Se abre la pestaña “Basic”.	En la parte superior izquierda se muestran los siguientes parámetros: - Número de serie. - Temperatura. - CPU activa. - Porcentaje de la CPU. - Temperatura externa.	OK
3	Se abre la pestaña “Basic”. Se escriben diferentes valores en el cuadro de texto que representa el número de serie.	Solo pueden introducirse números comprendidos entre 0 y 100000.	OK
4	La aplicación tiene conexión con el dispositivo. Se abre la pestaña “Basic”. Se inserta el número de serie del dispositivo en el cuadro de texto situado en la parte superior izquierda de la ventana. Se presiona el botón “C.OFF”.	La aplicación no tiene conexión con el dispositivo.	OK
5	Justo después de ejecutar la prueba anterior, se presiona el botón “C.ON”.	La aplicación tiene conexión con el dispositivo.	OK
6	La aplicación tiene conexión con el dispositivo. Se abre la pestaña “Basic”. Se presiona el botón “A.OFF”.	La aplicación no tiene conexión con los dispositivos dentro del rango del número de serie introducido.	OK
7	Justo después de ejecutar la prueba anterior, se presiona el botón “A.ON”.	La aplicación tiene conexión con los dispositivos dentro del rango del número de serie introducido.	OK

Tabla 6.2: Pruebas de la pestaña Basic.

Capítulo 7

Conclusiones y trabajo futuro

7.1. Conclusiones

En este apartado, debemos plantearnos si los objetivos propuestos al comienzo del proyecto se han alcanzado.

- ✓ Se ha conseguido la funcionalidad requerida, desarrollar un conjunto de herramientas que permiten la configuración remota de dispositivos UAV.
- ✓ Se ha elaborado una librería de comunicaciones que implementa el ICD y se ha integrado adecuadamente en Visionair y ha hecho un uso correcto del driver de comunicaciones.
- ✓ La aplicación "Device Config Tool" cubre todas las necesidades básicas para la configuración de los diversos dispositivos.
- ✓ Se ha conseguido encapsular tanto la aplicación como la librería de comunicaciones en una DLL que se ha integrado correctamente en Visionair.
- ✓ La herramienta desarrollada deja abierta la posibilidad de ampliación de funcionalidades, además, su estructura permite la reutilización de código en caso de la adicción de nuevos requisitos.

También debemos analizar las aportaciones del proyecto. La investigación sobre el uso de las diferentes plataformas ha proporcionado a la estudiante conocimientos sobre:

- ✓ La utilización del entorno de trabajo Microsoft Visual Studio, junto con el aprendizaje de la programación orientada a objetos basada en C# y el diseño de interfaces mediante WPF.
- ✓ También se han adquirido conocimientos sobre el uso de repositorios Git.
- ✓ Por último, se han adquirido conocimientos sobre el funcionamiento del mundo de los drones, desde su preparación hasta su puesta en marcha.

7.2. Trabajo futuro

En este apartado, se enumeran las posibles líneas futuras que podría adoptar la aplicación, debido a que el proyecto completo es mucho más amplio e incluye muchas más funcionalidades:

- Ampliar los idiomas que admite la aplicación.
- Ampliar las funciones de la aplicación de manera que admita la configuración de otros dispositivos, como, por ejemplo, el POLAR.
- Implementar una extensión para desarrolladores que incluya funciones que permitan la configuración avanzada de los dispositivos.
- Implementar una extensión que permita la carga de software en los dispositivos.
- Implementar una extensión que permita cargar un archivo de extensión ".uav" con el log de la ruta realizada por el dispositivo, con la finalidad de extraer diferentes parámetros y permitir la conversión de este log en un archivo con extensión ".txt".
- Implementar una extensión que permita la configuración de la radio Ground Control Station (GCS).

Bibliografía

[1] *Vehículo aéreo no tripulado*. 2016. URL: https://es.wikipedia.org/wiki/Veh%C3%ADculo_aÃľreo_no_tripulado.

[2] *Informe: Rusia inicia misiones de reconocimiento con drones en Siria*. 2015. URL: http://www.hispantv.com/newsdetail/siria/57918/rusia-siria-drones-reconocimiento.

[3] *Los 14 usos de drones que seguro no conocías*. 2014. URL: http://agencia.donweb.com/los-14-usos-de-drones-que-seguro-no-conocias/.

[4] *DHL comienza a probar repartir paquetes con drones en Alemania*. 2014. URL: http://es.gizmodo.com/dhl-comienza-a-repartir-paquetes-mediante-drones-en-ale-1639852203.

[5] *About UAVN*. 2016. URL: http://www.uavnavigation.com/company/about-uavn.

[6] *Todos los aviones blanco españoles vuelan con el AP04 de UAV Navigation*. 2010. URL: http://www.onemagazine.es/noticia/2441/sin-especificar/todos-los-aviones-blanco-espanoles-vuelan-con-el-ap04-de-uav-navigation.html.

[7] *UAV Navigation reconocida "Mejor empresa Tecnológica.^en la II edición del Spain Startup and Investor Summit 2013*. 2013. URL: http://www.defensa.com/frontend/defensa/uav-navigation-reconocida-mejor-empresa-tecnologica-ii-edicion-vn10487-vst241.

[8] *UAV Navigation contribuye a la integración de drones en el espacio aéreo*. 2015. URL: http://observatorio.cisde.es/archivo/uav-navigation-contribuye-a-la-integracion-de-drones-en-el-espacio-aereo/.

[9] *The Martin Jetpack Project*. 2015. URL: http://www.uavnavigation.com/company/news-and-events/martin-jetpack-project.

[10] *Autopilotos*. 2015. URL: http://www.uavnavigation.com/support/kb/autopilots.

[11] *Visionair*. 2016. URL: http://www.uavnavigation.com/support/kb/software/visionair.

[12] *EventArgs Class*. 2016. URL: https://msdn.microsoft.com/en-us/library/system.eventargs(v=vs.100).aspx.

[13] *The MVVM Pattern*. 2012. URL: https://msdn.microsoft.com/en-us/library/hh848246.aspx.

[14] *MVC y MVVM*. 2012. URL: https://www.adictosaltrabajo.com/tutoriales/zk-mvc-mvvm/.

[15] *The MacTeX-2015 Distribution*. 2015. URL: http://tug.org/mactex/mactex-download.html.

[16] *Visual Studio Community*. 2016. URL: https://www.visualstudio.com/es-es/products/visual-studio-community-vs.aspx.

[17] *C Sharp*. 2016. URL: https://msdn.microsoft.com/es-es/library/kx37x362.aspx.

[18] *WPF Tutorial.net*. 2011. URL: http://www.wpftutorial.net.

[19] *Adobe Photoshop CC*. 2016. URL: http://www.adobe.com/es/products/photoshop.html.

[20] *TUTORIAL DE GANTTPROJECT*. 2015. URL: http://ganttproject-equipo3.blogspot.com.es.

[21] *Bitbucket*. 2016. URL: https://bitbucket.org.

[22] *Sublime Text*. 2016. URL: https://www.sublimetext.com.

[23] *INotifyPropertyChanged Interface*. 2016. URL: https://msdn.microsoft.com/en-us/library/system.componentmodel.inotifypropertychanged(v=vs.110).aspx.

Apéndices

Apéndice A

Clases del paquete de recursos

Aquí se muestran las clases del paquete de recursos con sus métodos y atributos.

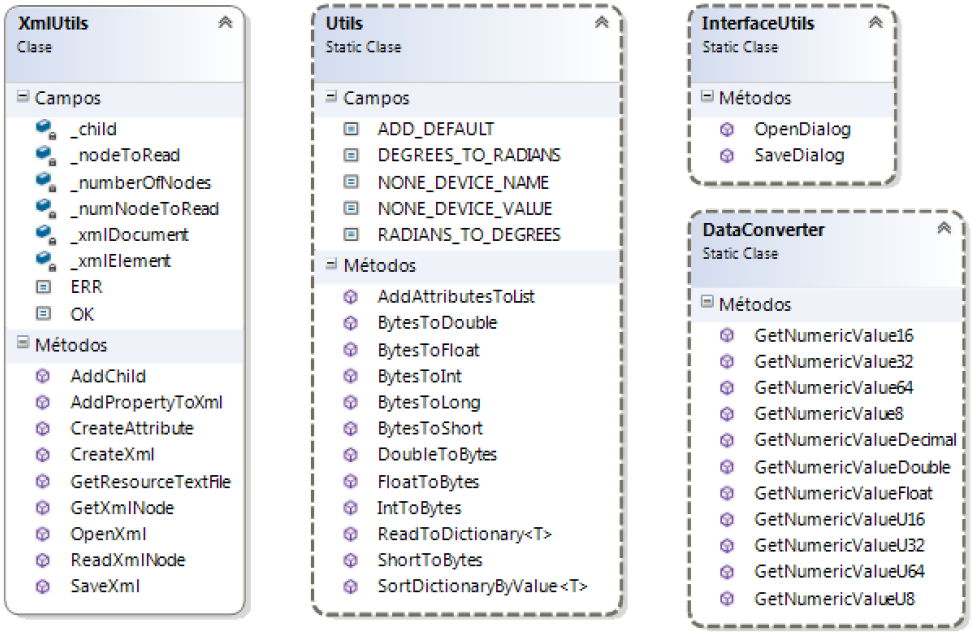


Figura A.1: Clases del paquete Recursos.

Apéndice B

Clase del tipo EventArgs

Este es el formato de las clases que definen los mensajes, contiene atributos por cada parámetro contenido en el mensaje, propiedades que proporcionan estos valores y un constructor de la clase, el cual recibe como parámetros los valores extraídos del mensaje.

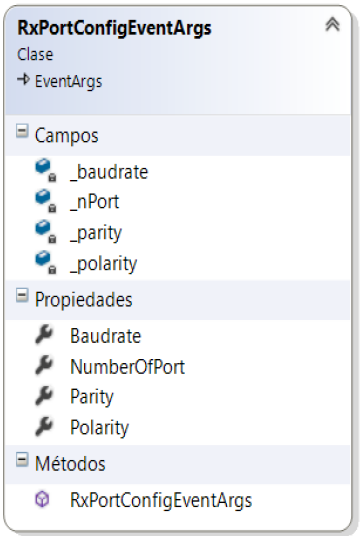


Figura B.1: Clase RxPortConfigEventArgs.

Apéndice C

Paquetes Device Config Tool

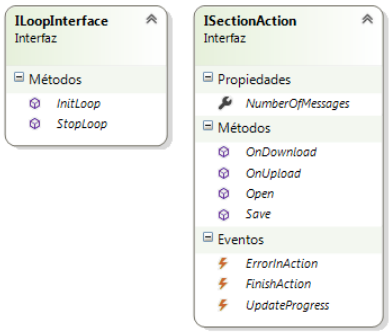


Figura C.1: Conjunto de interfaces.

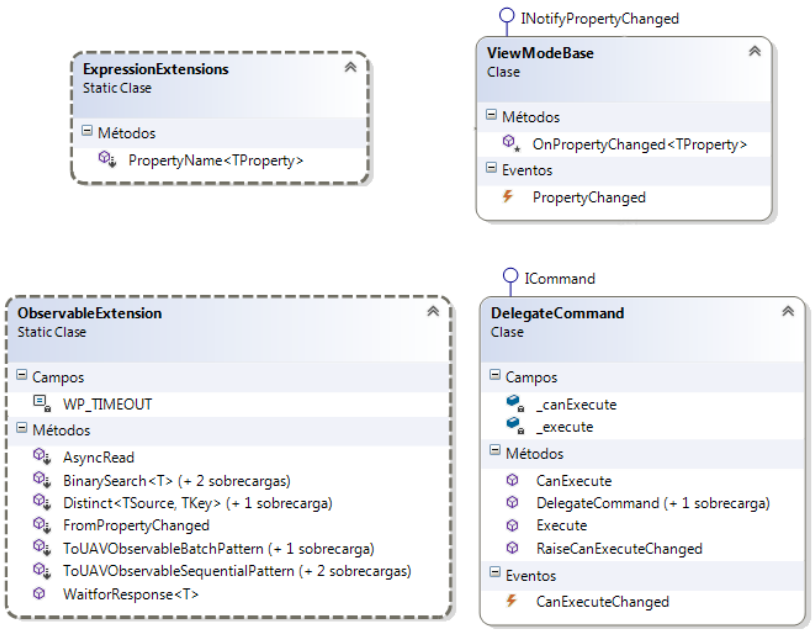


Figura C.2: Clases de soporte.

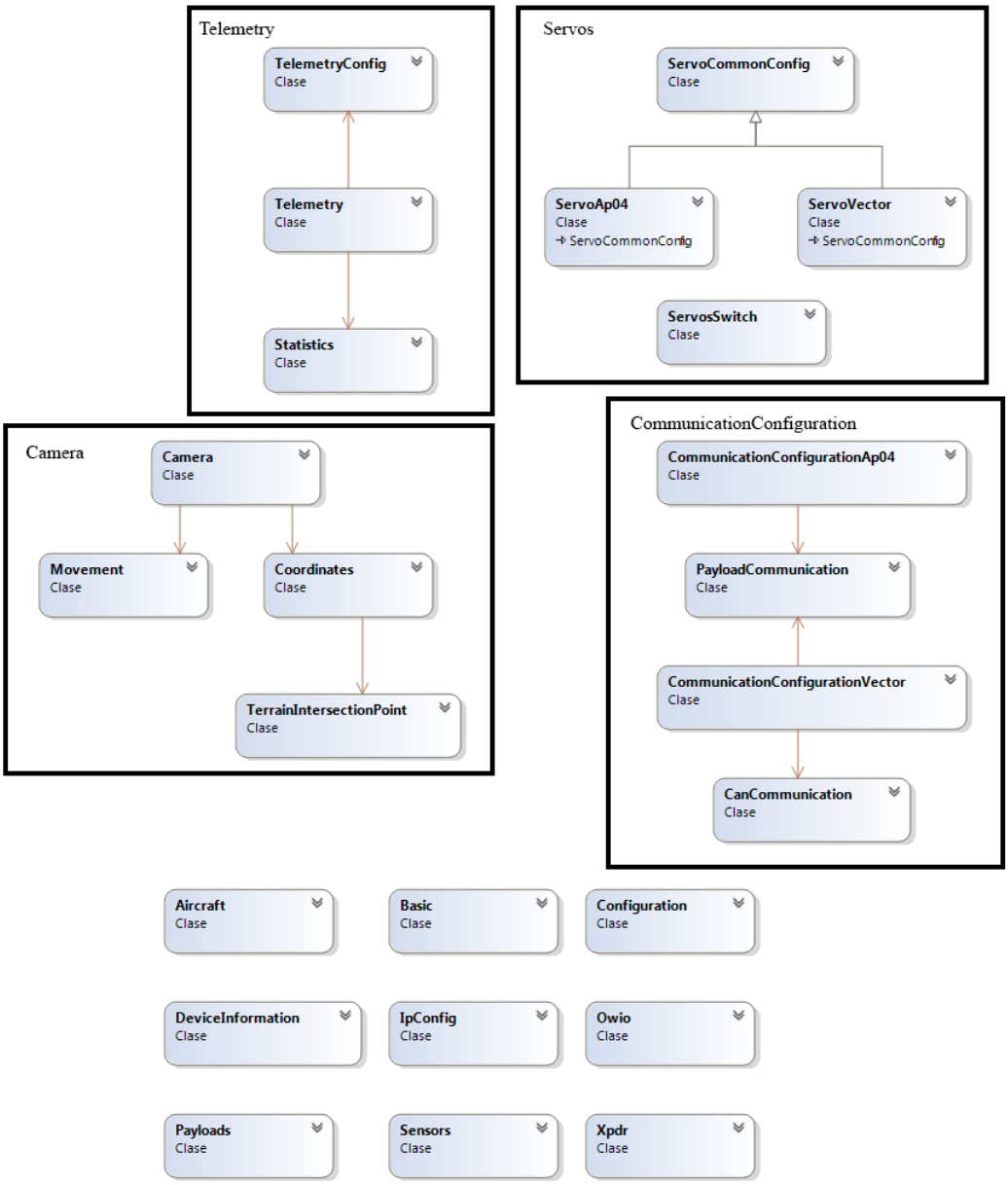


Figura C.3: Diagrama de clases de los modelos.

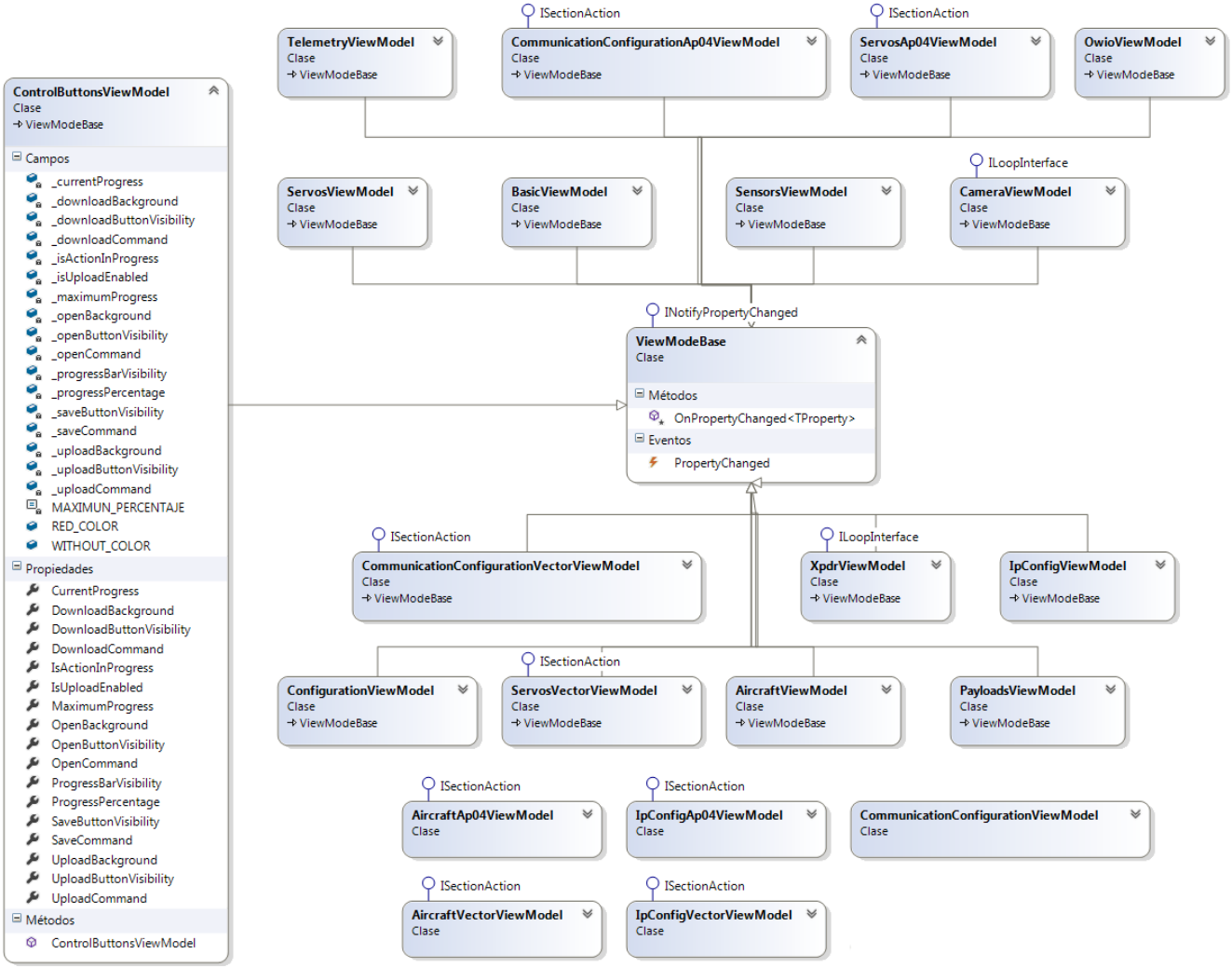


Figura C.4: Diagrama del conjunto de ViewModels.

Apéndice D

Ejemplos de ViewModel

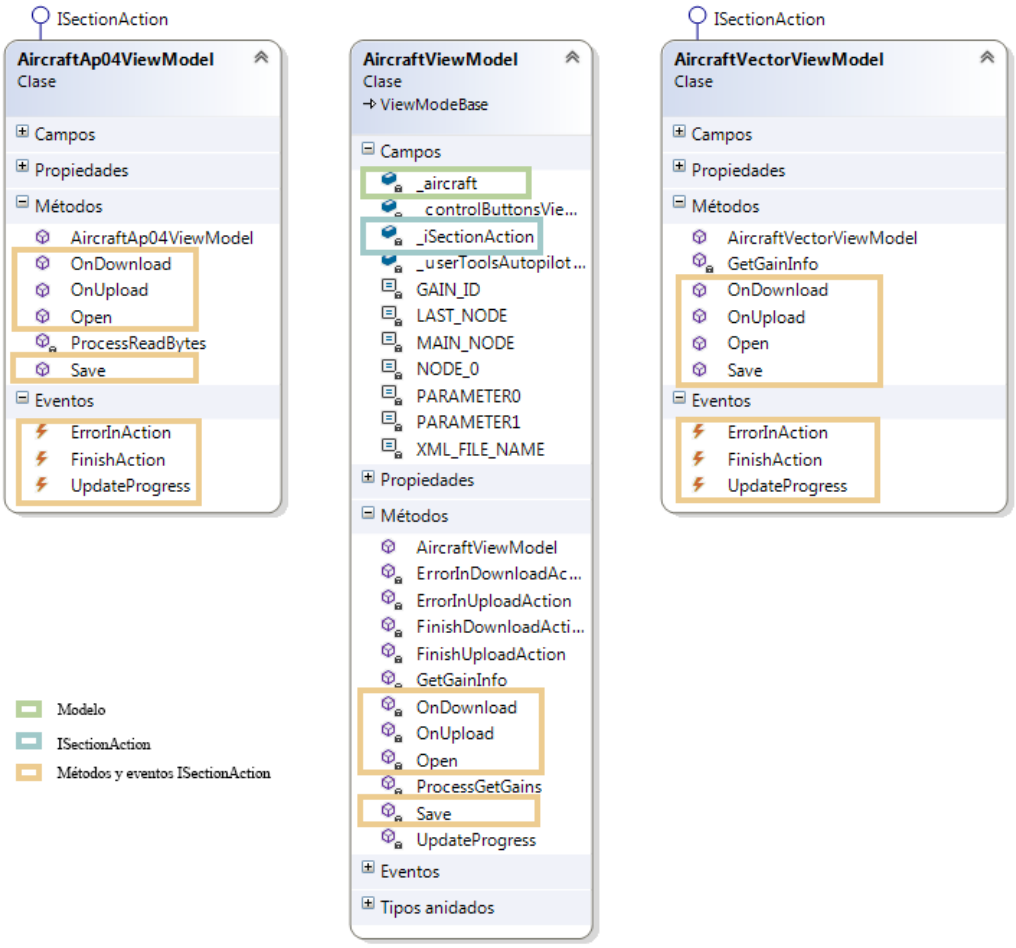


Figura D.1: Esquema del ViewModel para la pestaña Aircraft.

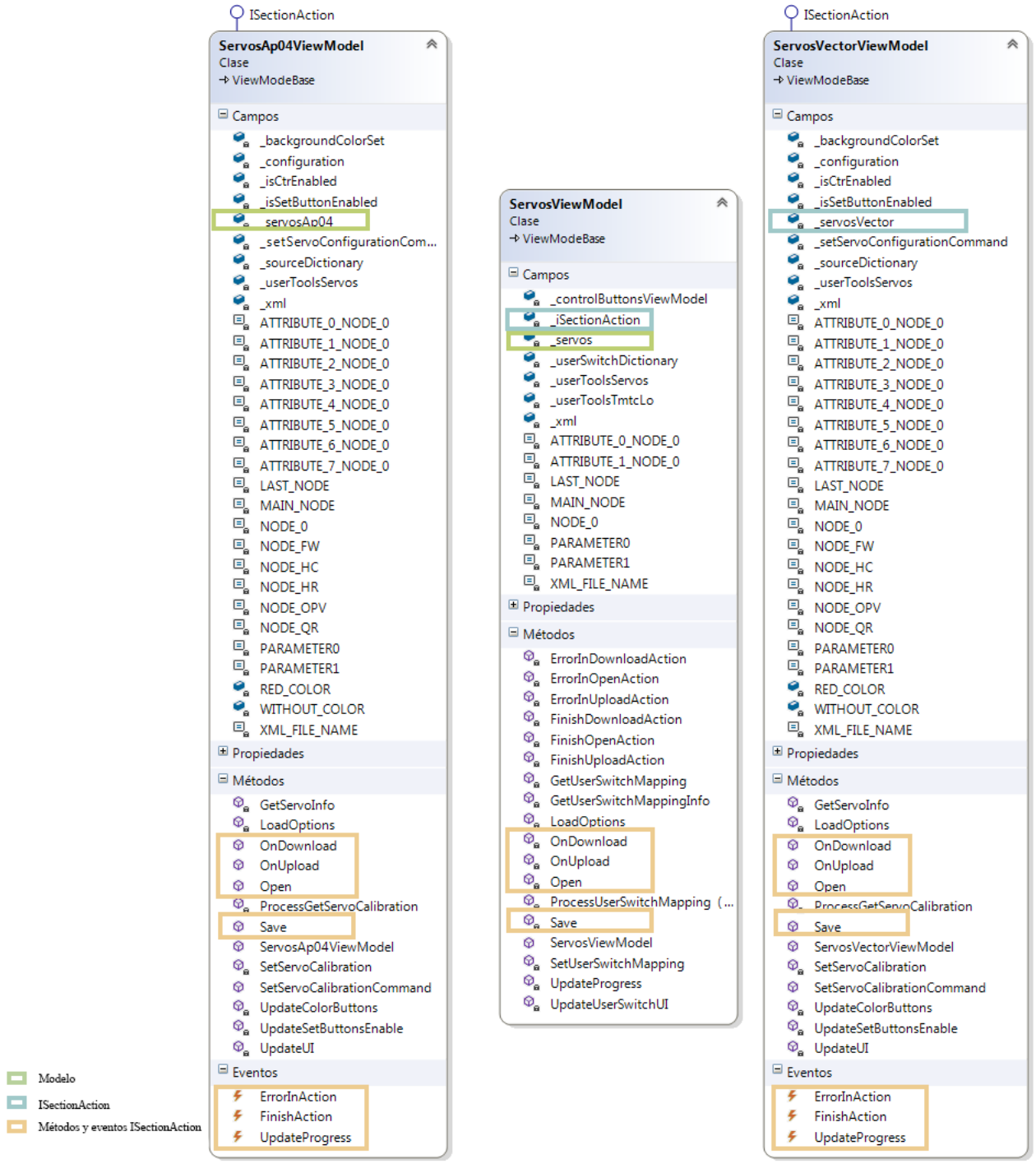


Figura D.2: Esquema del ViewModel para la pestaña Servos.

Apéndice E

Pruebas de las pestañas de la aplicación

E.1. Pestaña Aircraft

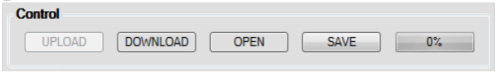
Nº	Descripción	Resultado esperado	Test
1	Se abre la pestaña "Aircraft".	En la parte de abajo de la pestaña se muestra un grupo de botones como este: 	OK
2	Se abre la pestaña "Aircraft". Se presiona el botón "Download".	Se muestra el "Aircraft ID".	OK

Tabla E.1: Pruebas de la pestaña Aircraft.

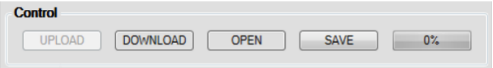
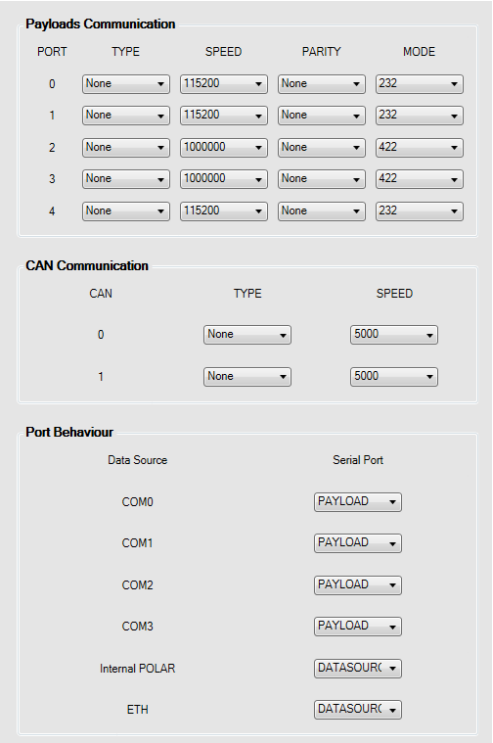
E.2. Pestaña Cfg

Nº	Descripción	Resultado esperado	Test
1	Se abre la pestaña "Cfg".	En la parte inferior de la ventana aparece el siguiente grupo de botones: 	OK
2	Se abre la pestaña "Cfg". Se presiona el botón "Download".	La siguiente información aparece en "Part Number": - Tipo de dispositivo. - Versión del software. - CRC. - Versión del estimador.	OK

Nº	Descripción	Resultado esperado	Test
3	Se abre la pestaña “Cfg”. Se presiona el botón “Polar PGM”.	Si todo es correcto, el POLAR se pondrá en modo PGM.	OK
4	Se abre la pestaña “Cfg”. Se presiona el botón “Polar PGM”.	Si ha ocurrido algún error, el modo del POLAR no variará y el botón se pondrá de color rojo.	OK

Tabla E.2: Pruebas de la pestaña Cfg.

E.3. Pestaña COMM Cfg

Nº	Descripción	Resultado esperado	Test
1	Se abre la pestaña “COMM Cfg”.	En la parte inferior de la ventana aparece un grupo de botones como el siguiente: 	OK
2	Se abre la pestaña “COMM Cfg”.	Si un VECTOR está conectado, la interfaz se muestra como la siguiente: 	OK

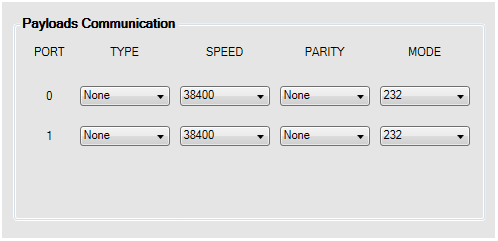
Nº	Descripción	Resultado esperado	Test
3	Se abre la pestaña "COMM Cfg".	Si un AP04 está conectado, la interfaz se muestra como la siguiente: 	OK
4	Se abre la carpeta "Documents/Visionair/User Tools" y se edita el archivo "CommConfigVector.xml" o el archivo "CommConfigAp04.xml", dependiendo del tipo de dispositivo conectado. Se ejecuta la aplicación. Se abre la pestaña "COMM Cfg".	Las opciones de los menús desplegables coinciden con las opciones del XML modificado.	OK

Tabla E.3: Pruebas de la pestaña COMM Cfg.

E.4. Pestaña IpCfg

Nº	Descripción	Resultado esperado	Test
1	Se abre la pestaña "IpCfg".	En la parte inferior de la ventana aparece el siguiente grupo de botones: 	OK
2	Se abre la pestaña "IpCfg". Se escriben diferentes valores en los cuadros de texto que representan la IP local.	Solo pueden introducirse números comprendidos entre 0 y 255.	OK
3	Se abre la pestaña "IpCfg". Se escriben diferentes valores en los cuadros de texto que representan las IP destino.	Solo pueden introducirse números comprendidos entre 0 y 255.	OK
4	Se abre la pestaña "IpCfg". Selecciona la opción "Broadcast".	Las cajas de texto para definir las IP de destino han sido deshabilitadas.	OK
5	Se abre la pestaña "IpCfg". Se escriben diferentes valores en el cuadro de texto que representa el puerto.	Solo pueden introducirse números comprendidos entre 0 y 32767.	OK

Tabla E.4: Pruebas de la pestaña IpCfg.

E.5. Pestaña GPIO

Nº	Descripción	Resultado esperado	Test
1	Se abre la carpeta "Documents/Visionair/User Tools" y se edita el fichero "Owio.xml". Se ejecuta la aplicación. Se abre la pestaña "GPIO".	Las opciones de los menús desplegables se corresponden con las introducidas en el XML.	OK
2	Se abre la pestaña "GPIO".	En la parte inferior de la ventana aparece el siguiente grupo de botones: 	OK

Tabla E.5: Pruebas de la pestaña GPIO.

E.6. Pestaña TM/TC

Nº	Descripción	Resultado esperado	Test
1	Se abre la pestaña "TM/TC".	En la parte inferior de la ventana aparece el siguiente grupo de botones: 	OK
2	Se abre la pestaña "TM/TC". Se selecciona alguna de las opciones, "Secondary ADS", "Extended Cmd/Err Data" o "Engine Monitor". Se selecciona "Target Mini Telem".	Todas las opciones están deseleccionadas salvo "Target Mini Telem".	OK
3	Se abre la pestaña "TM/TC". Se selecciona la opción "Target Mini Telem". Se deselecciona "Target Mini Telem".	Todas las opciones están deseleccionadas salvo "Secondary ADS".	OK
4	Se abre la pestaña "TM/TC". Se selecciona la opción "Target Mini Telem". Se selecciona cualquier otra opción.	La opción "Target Mini Telem" se ha deseleccionado.	OK
5	Se abre la pestaña "TM/TC". Se escriben diferentes valores en el cuadro de texto que representa el valor "LOS time".	Solo pueden introducirse números comprendidos entre 5 y 999.	OK

Tabla E.6: Pruebas de la pestaña TM/TM.

E.7. Pestaña Sns

Nº	Descripción	Resultado esperado	Test
1	Se abre la pestaña "Sns".	Se pueden monitorizar los valores de los ADC.	OK
2	Se abre la pestaña "Sns".	Se pueden monitorizar los valores de los sensores.	OK

Tabla E.7: Pruebas de la pestaña Sns.

E.8. Pestaña Payloads

Nº	Descripción	Resultado esperado	Test
1	Se abre la pestaña "Payloads".	Aparecen dos pestañas, "Camera" y "XPDR", en este orden.	OK

Tabla E.8: Pruebas de la pestaña Payloads.

E.9. Pestaña Cam

Nº	Descripción	Resultado esperado	Test
1	Se abre la pestaña "Payloads/Cam".	Si la opción "Payload Telemetry" de la pestaña "Cfg" está desactivada, aparecerá la siguiente advertencia: "WARNING: The option "Payload Telemetry" is not active and no information is received.".	OK
2	Se abre la pestaña "Payloads/Cam".	Si la opción "Fixed" de la pestaña "Cfg" está activada, aparecerá la siguiente advertencia: "WARNING: The option "Payload Fixed" is active and the changes won't take effect.".	OK
3	Se abre la pestaña "Payloads/Cam".	Se muestran, en la parte superior de la ventana, dos dibujos que representas los valores de pan y tilt de la cámara conectada.	OK
4	Se abre la pestaña "Payloads/Cam". Se selecciona un modo.	Si no ha ocurrido ningún error, el modo de la cámara será actualizado en el dispositivo.	OK

Nº	Descripción	Resultado esperado	Test
5	Se abre la pestaña "Payloads/Cam". Se selecciona un modo.	Si ha ocurrido algún error, el radio button correspondiente al modo se pondrá de color rojo.	OK
6	Se abre la pestaña "Payloads/Cam". Se selecciona un modo. Se modifican los parámetros del modo. Pulsa el botón "Set".	Si no ha ocurrido ningún error, la configuración será actualizada en el dispositivo.	OK
7	Se abre la pestaña "Payloads/Cam". Se selecciona un modo. Se modifican los parámetros del modo. Pulsa el botón "Set".	Si ha ocurrido algún error, el botón "Set" correspondiente al modo se pondrá de color rojo.	OK
8	Se abre la pestaña "Payloads/Cam". Se escriben diferentes valores en el cuadro de texto que representa el valor pan en el modo Aircraft.	Solo pueden introducirse números comprendidos entre -360 y 3600.	OK
9	Se abre la pestaña "Payloads/Cam". Se escriben diferentes valores en el cuadro de texto que representa el valor tilt en el modo Aircraft.	Solo pueden introducirse números comprendidos entre -90 y 90.	OK
10	Se abre la pestaña "Payloads/Cam". Se escriben diferentes valores en el cuadro de texto que representa el valor heading.	Solo pueden introducirse números comprendidos entre -3.4E+38 y 3.4E+38.	OK
11	Se abre la pestaña "Payloads/Cam". Se escriben diferentes valores en el cuadro de texto que representa el valor elevation.	Solo pueden introducirse números comprendidos entre -3.4E+38 y 3.4E+38.	OK
12	Se abre la pestaña "Payloads/Cam". Se escriben diferentes valores en el cuadro de texto que representa el valor pan en el modo Inertial.	Solo pueden introducirse números comprendidos entre -32768 y 32767.	OK
13	Se abre la pestaña "Payloads/Cam". Se escriben diferentes valores en el cuadro de texto que representa el valor tilt en el modo Inertial.	Solo pueden introducirse números comprendidos entre -32768 y 32767.	OK
14	Se abre la pestaña "Payloads/Cam". Se escriben diferentes valores en el cuadro de texto que representa el valor latitude.	Solo pueden introducirse números comprendidos entre -32768 y 32767.	OK

Nº	Descripción	Resultado esperado	Test
15	Se abre la pestaña "Payloads/Cam". Se escriben diferentes valores en el cuadro de texto que representa el valor longitude.	Solo pueden introducirse números comprendidos entre -32768 y 32767.	OK
16	Se abre la pestaña "Payloads/Cam". Se escriben diferentes valores en el cuadro de texto que representa el valor altitude.	Solo pueden introducirse números comprendidos entre -32768 y 32767.	OK
17	Se abre la pestaña "Payloads/Cam". Se define una configuración para el control de la imagen de la cámara. Se presiona el botón "Send Config".	Si no ha ocurrido ningún error, la configuración será actualizada en el dispositivo.	OK
18	Se abre la pestaña "Payloads/Cam". Se define una configuración para el control de la imagen de la cámara. Se presiona el botón "Send Config".	Si ha ocurrido algún error, la configuración no será actualizada en el dispositivo y el botón "Send Config" se pondrá de color rojo.	OK
19	Se abre la pestaña "Payloads/Cam". Se presiona el botón "Send Reset Flip Config".	Si no ha ocurrido ningún error, la configuración será actualizada en el dispositivo.	OK
20	Se abre la pestaña "Payloads/Cam". Se presiona el botón "Send Reset Flip Config".	Si ha ocurrido algún error, la configuración no será actualizada en el dispositivo y el botón "Send Reset Flip Config" se pondrá de color rojo.	OK
21	Se abre la pestaña "Payloads/Cam". Se ajusta un nivel de zoom.	Si no ha ocurrido ningún error, el valor de zoom será actualizado en el dispositivo.	OK
22	Se abre la pestaña "Payloads/Cam". Se ajusta un nivel de zoom.	Si ha ocurrido algún error, el valor de zoom no será actualizado en el dispositivo y el slider tomará el valor de zoom que actualmente tiene el dispositivo.	OK

Tabla E.9: Pruebas de la pestaña Cam.

E.10. Pestaña Servos

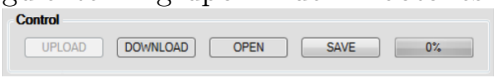
Nº	Descripción	Resultado esperado	Test
1	Se abre la pestaña "Servos".	En la parte inferior de la ventana aparece el siguiente grupo de botones: 	OK
2	Se abre la carpeta "Documents/Visionair/User Tools" y se edita el archivo "ServosVector.xml" o el archivo "ServosAp04.xml". Se ejecuta la aplicación. Se abre la pestaña "Servos".	Las opciones de los menús desplegables se corresponden con las introducidas en el XML.	OK
3	Se abre la pestaña "Servos". Se escriben diferentes valores en los cuadros de texto que representan el valor mínimo.	Solo pueden introducirse números comprendidos entre 0 y 1000.	OK
4	Se abre la pestaña "Servos". Se escriben diferentes valores en los cuadros de texto que representan el valor máximo.	Solo pueden introducirse números comprendidos entre 0 y 1000.	OK
5	Se abre la pestaña "Servos". Se escriben diferentes valores en los cuadros de texto que representan el valor central.	Solo pueden introducirse números comprendidos entre 0 y 1000.	OK
6	Se conecta un VECTOR. Se abre la pestaña "Servos". Se escriben diferentes valores en los cuadros de texto que representan el valor mínimo comandado.	Solo pueden introducirse números comprendidos entre -1 y 0.	OK
7	Se conecta un VECTOR. Se abre la pestaña "Servos". Se escriben diferentes valores en los cuadros de texto que representan el valor máximo comandado.	Solo pueden introducirse números comprendidos entre 0 y 1.	OK
8	Se conecta un AP04. Se abre la pestaña "Servos". Se escriben diferentes valores en los cuadros de texto que representan el valor de rango.	Solo pueden introducirse números comprendidos entre 0 y 360.	OK

Tabla E.10: Pruebas de la pestaña Servos.

E.11. Pestaña XPDR

Nº	Descripción	Resultado esperado	Test
1	Se abre la carpeta "Documents/-Visionair/User Tools" y se edita el archivo "XPDR.xml". Se ejecuta la aplicación. Se abre la pestaña "Payloads/XPDR".	Las opciones de los desplegables son las definidas en el XML.	OK
2	Se abre la pestaña "Payloads/XPDR". En la parte de "Installation" se presiona el botón "Get".	Si todo es correcto, la configuración del transponedor se mostrará en la interfaz.	OK
3	Se abre la pestaña "Payloads/XPDR". En la parte de "Installation" se presiona el botón "Get".	Si ha ocurrido algún error, la configuración no se mostrará y el botón "Get" se pondrá de color rojo.	OK
4	Se abre la pestaña "Payloads/XPDR". Se seleccionan unos valores de instalación. Presiona el botón "Set".	Si todo es correcto, la configuración del dispositivo será modificada.	OK
5	Se abre la pestaña "Payloads/XPDR". Se seleccionan unos valores de instalación. Se presiona el botón "Set".	Si ha habido algún error, la configuración en el dispositivo no será modificada y el botón "Set" se pondrá de color rojo.	OK
6	Se abre la pestaña "Payloads/XPDR". Se determina un identificador de vuelo. Se presiona el botón "Set".	Si todo es correcto, el identificador será actualizado en el dispositivo.	OK
7	Se abre la pestaña "Payloads/XPDR". Se determina un identificador de vuelo. Se presiona el botón "Set".	Si ha habido algún error, el identificador no será actualizado en el dispositivo y el botón "Set" se pondrá de color rojo.	OK
8	Se abre la pestaña "Payloads/XPDR". Se define un código para el transponedor. Se presiona el botón "Set Code".	Si todo es correcto, el código será actualizado en el dispositivo.	OK

Nº	Descripción	Resultado esperado	Test
9	Se abre la pestaña "Payloads/XPDR". Se define un código para el transponedor. Se presiona el botón "Set Code".	Si ha habido algún error, el código no será actualizado en el dispositivo y el botón "Set Code" se pondrá de color rojo.	OK
10	Se abre la pestaña "Payloads/XPDR". Se define un modo para el transponedor. Se presiona el botón "Set Mode".	Si todo es correcto, el modo será actualizado en el dispositivo	OK
11	Se abre la pestaña "Payloads/XPDR". Se define un modo para el transponedor. Se presiona el botón "Set Mode".	Si ha habido algún error, el código no será actualizado en el dispositivo y el botón "Set Mode" se pondrá de color rojo.	OK
12	Se abre la pestaña "Payloads/XPDR". Se define una altitud para el transponedor. Se presiona el botón "Set Alt".	Si todo es correcto, la altitud será actualizada en el dispositivo.	OK
13	Se abre la pestaña "Payloads/XPDR". Se define una altitud para el transponedor. Se presiona el botón "Set Alt".	Si ha habido algún error, la altitud no será actualizada en el dispositivo y el botón "Set Alt" se pondrá de color rojo.	OK
14	Se abre la pestaña "Payloads/XPDR". Se presiona el botón "Set Ident".	Si todo es correcto, la configuración en el dispositivo será modificada.	OK
15	Se abre la pestaña "Payloads/XPDR". Se presiona el botón "Set Ident".	Si ha habido algún error, el botón "Set Ident" se pondrá de color rojo.	OK
16	Se abre la pestaña "Payloads/XPDR". Se activa la opción "Poll".	En la interfaz se muestra información sobre el transponedor.	OK
17	Se abre la pestaña "Payloads/XPDR". Se desactiva la opción "Poll".	En la interfaz no se muestra información del transponedor.	OK

Tabla E.11: Pruebas de la pestaña XPDR.

Apéndice F

Vista de la aplicación

A continuación se muestran las imágenes de cada una de las pestañas de la aplicación final.

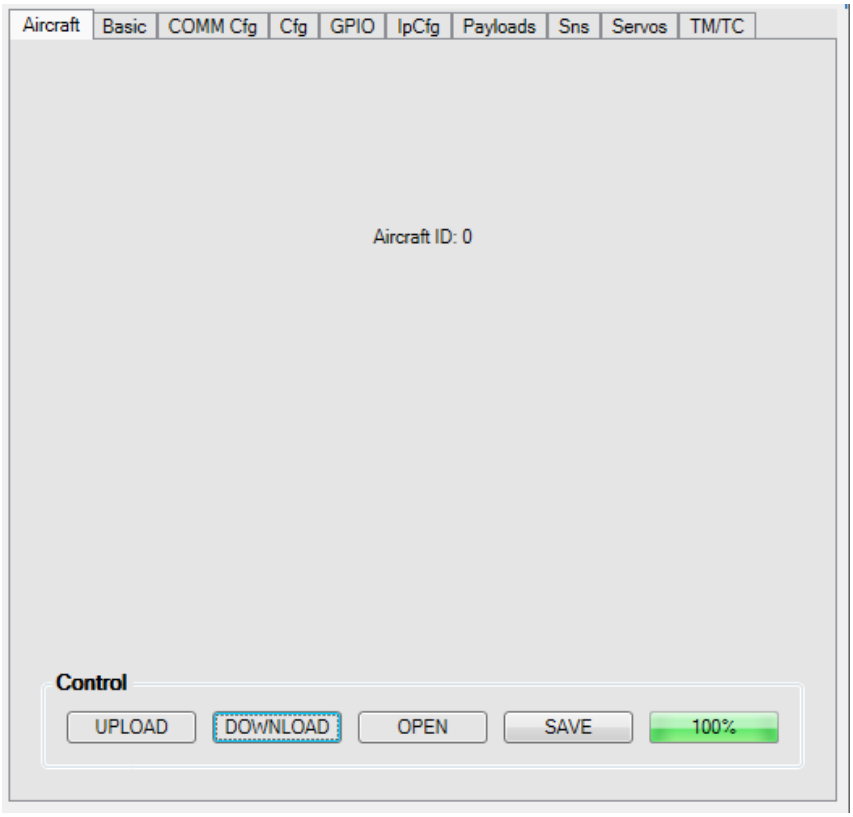


Figura F.1: Vista de la pestaña de ganancias.

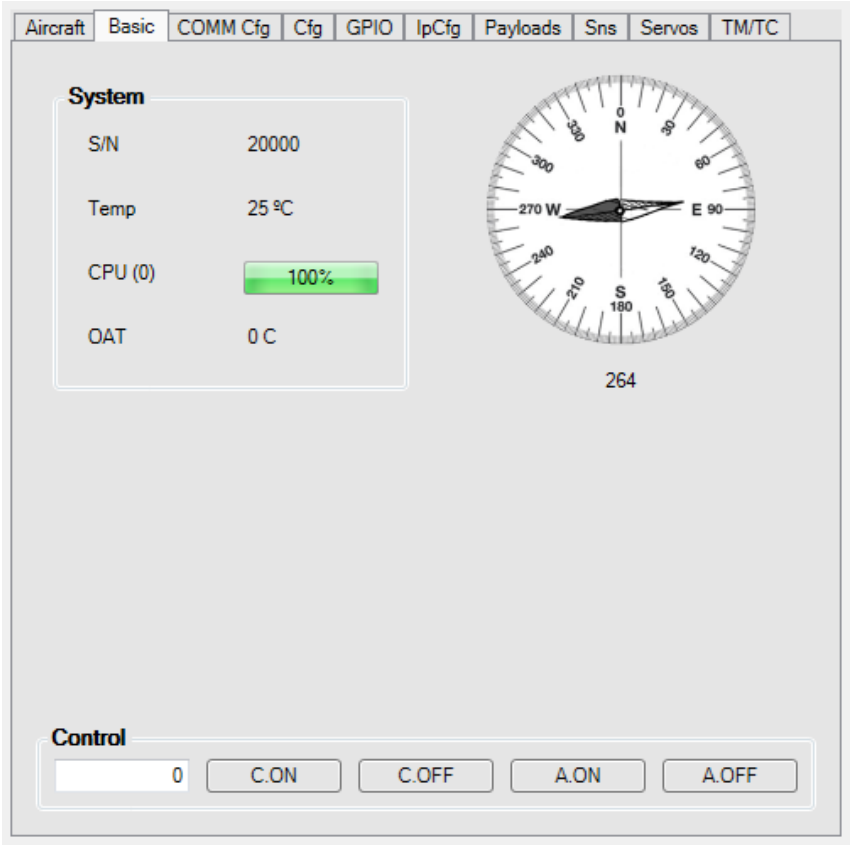


Figura F.2: Vista de la pestaña que muestra datos básicos del dispositivo.

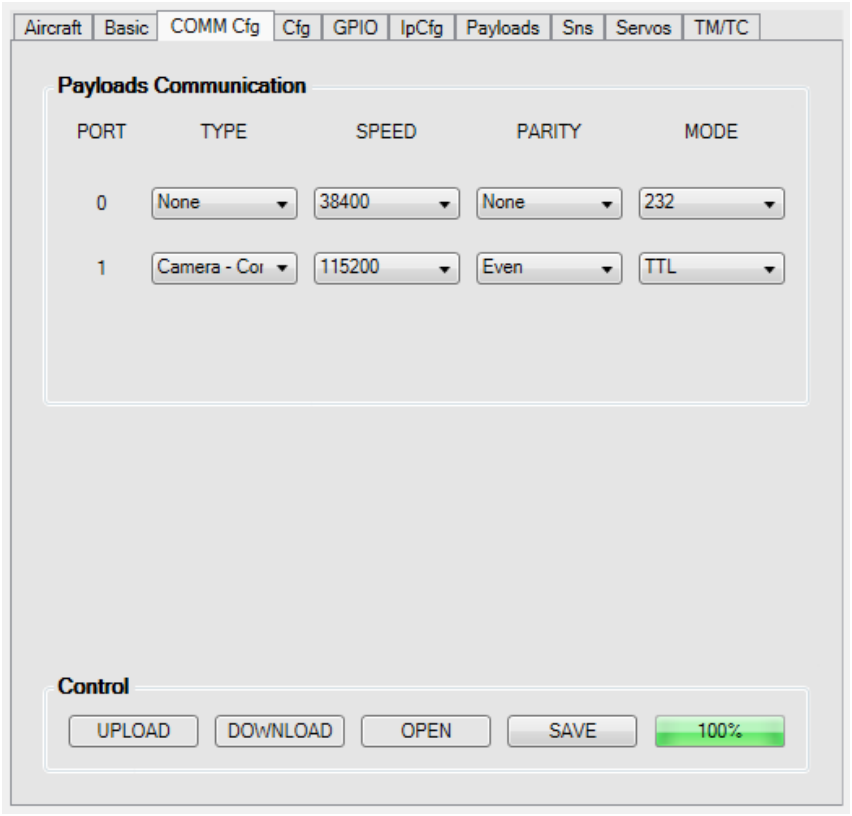


Figura F.3: Vista de la pestaña de configuración de las comunicaciones para un AP04.

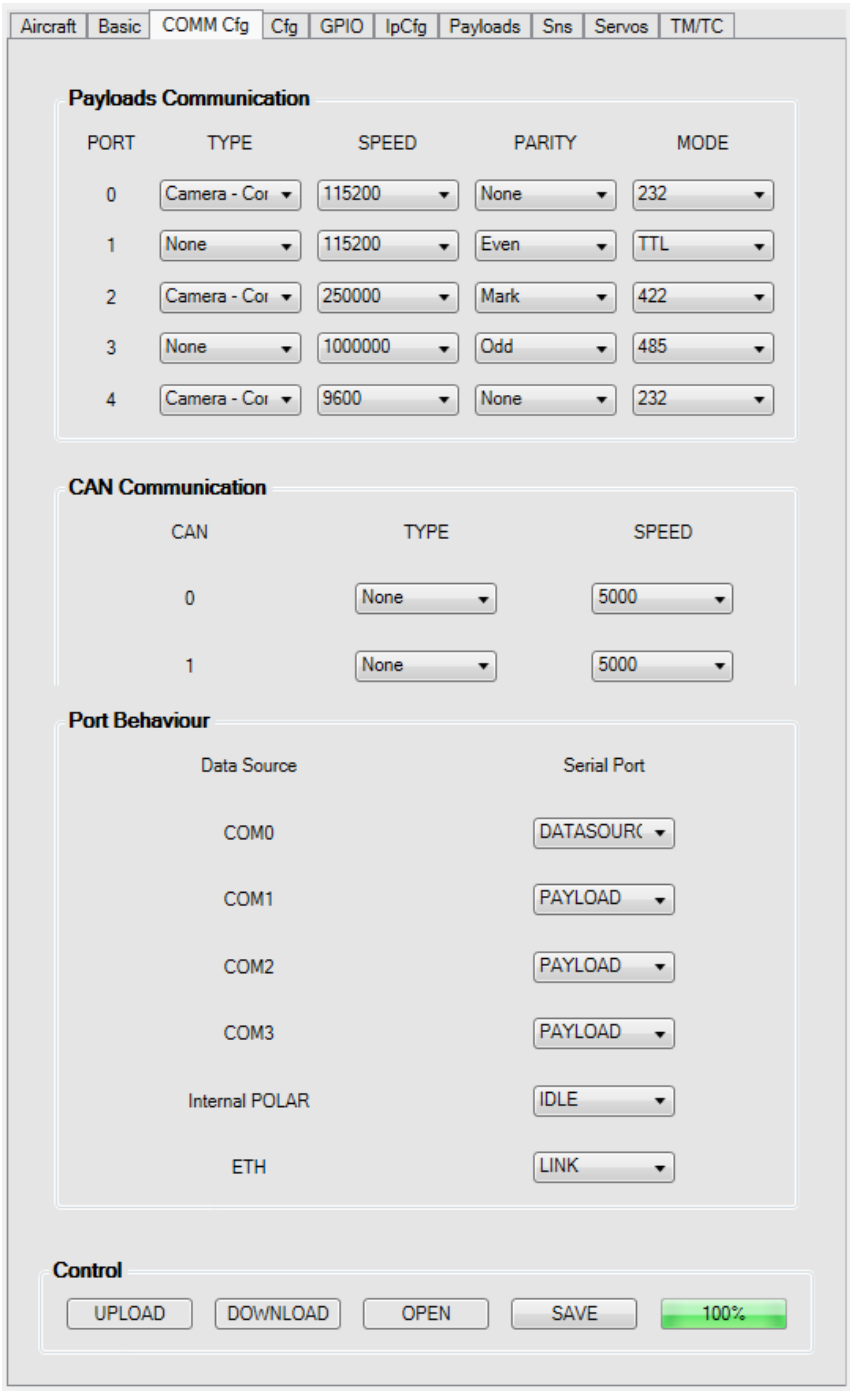


Figura F.4: Vista de la pestaña de configuración de las comunicaciones para un VECTOR.

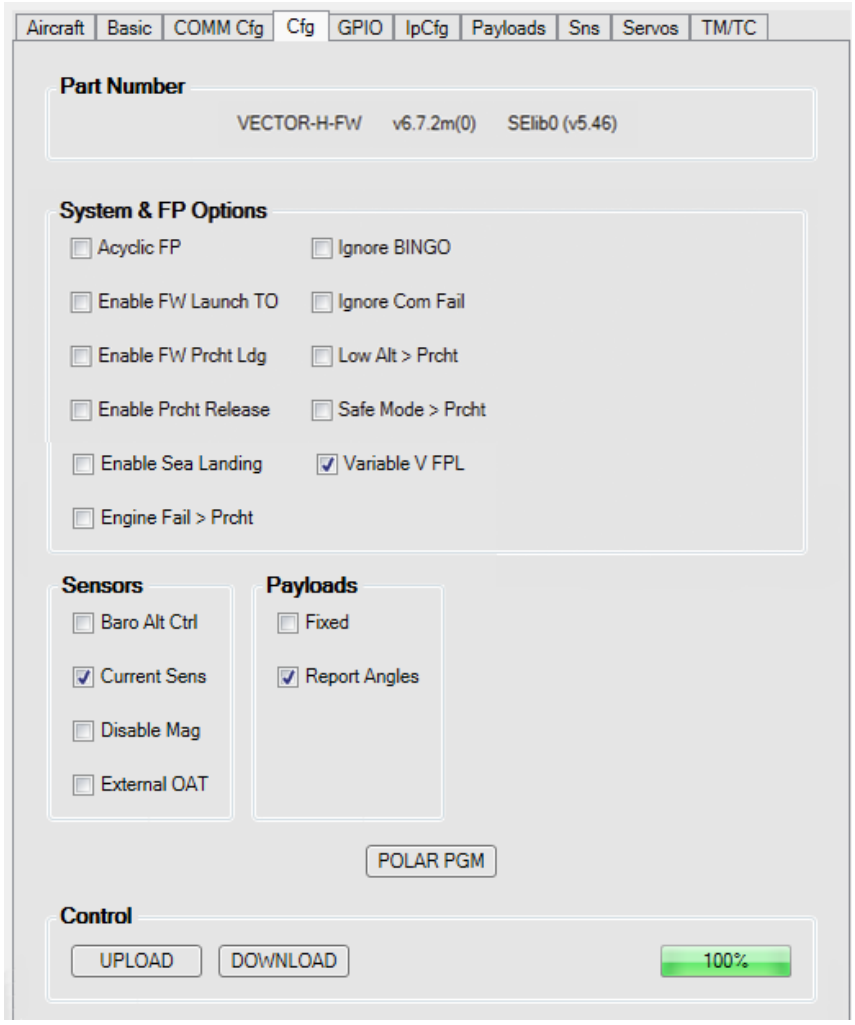


Figura F.5: Vista de la pestaña de configuración de planes de vuelo, sensores y payloads.

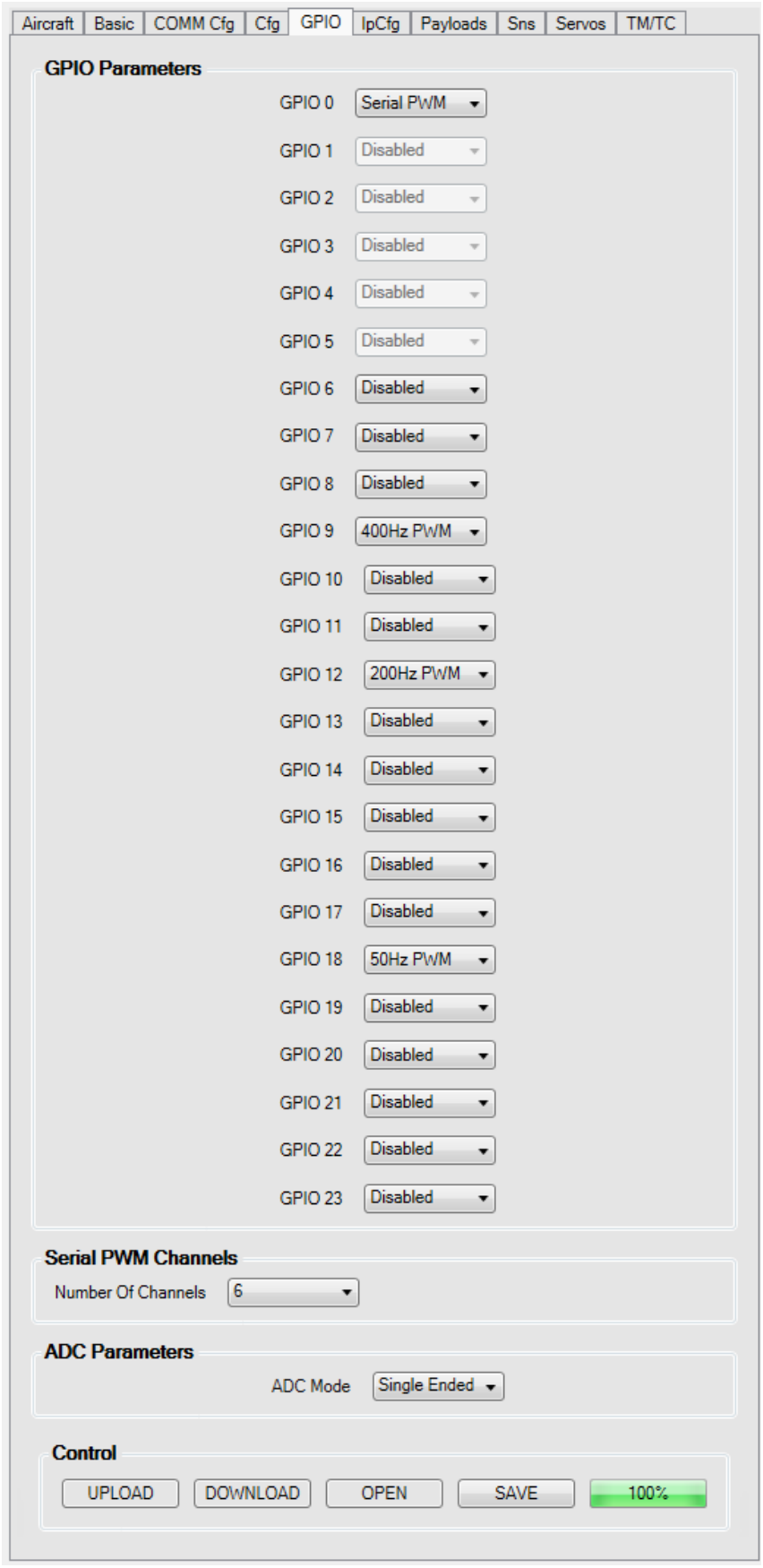


Figura F.6: Vista de la pestaña de configuración de los pines GPIO.

Figura F.7: Vista de la pestaña de configuración de la cámara.

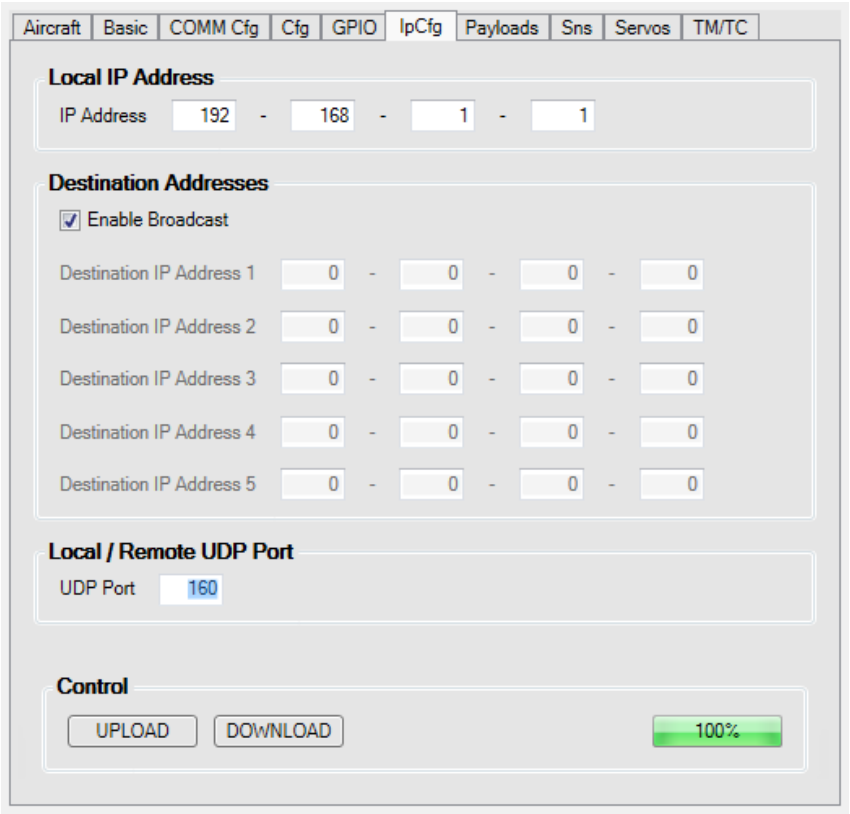


Figura F.8: Vista de la pestaña de configuración de las IP.

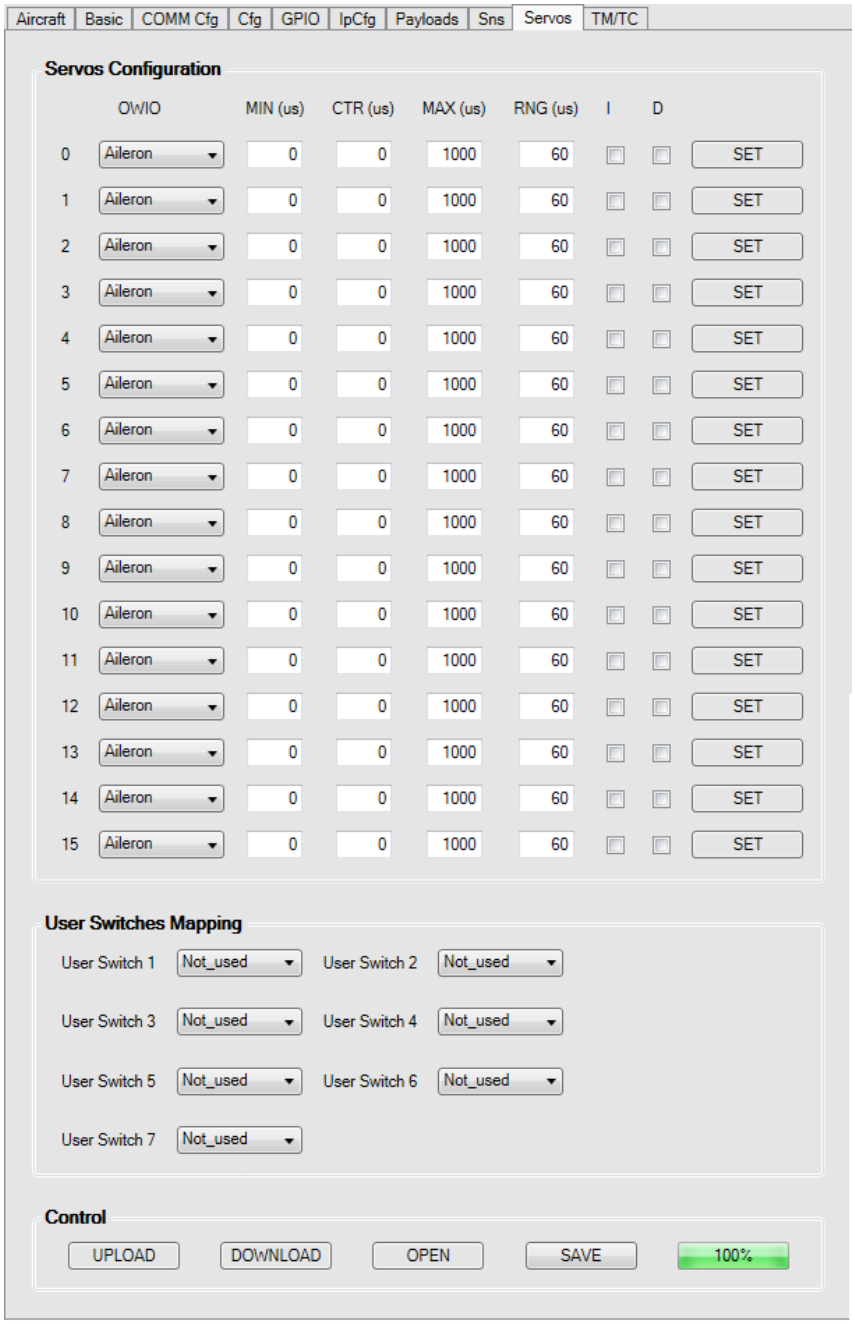


Figura F.9: Vista de la pestaña de configuración de servos para AP04.

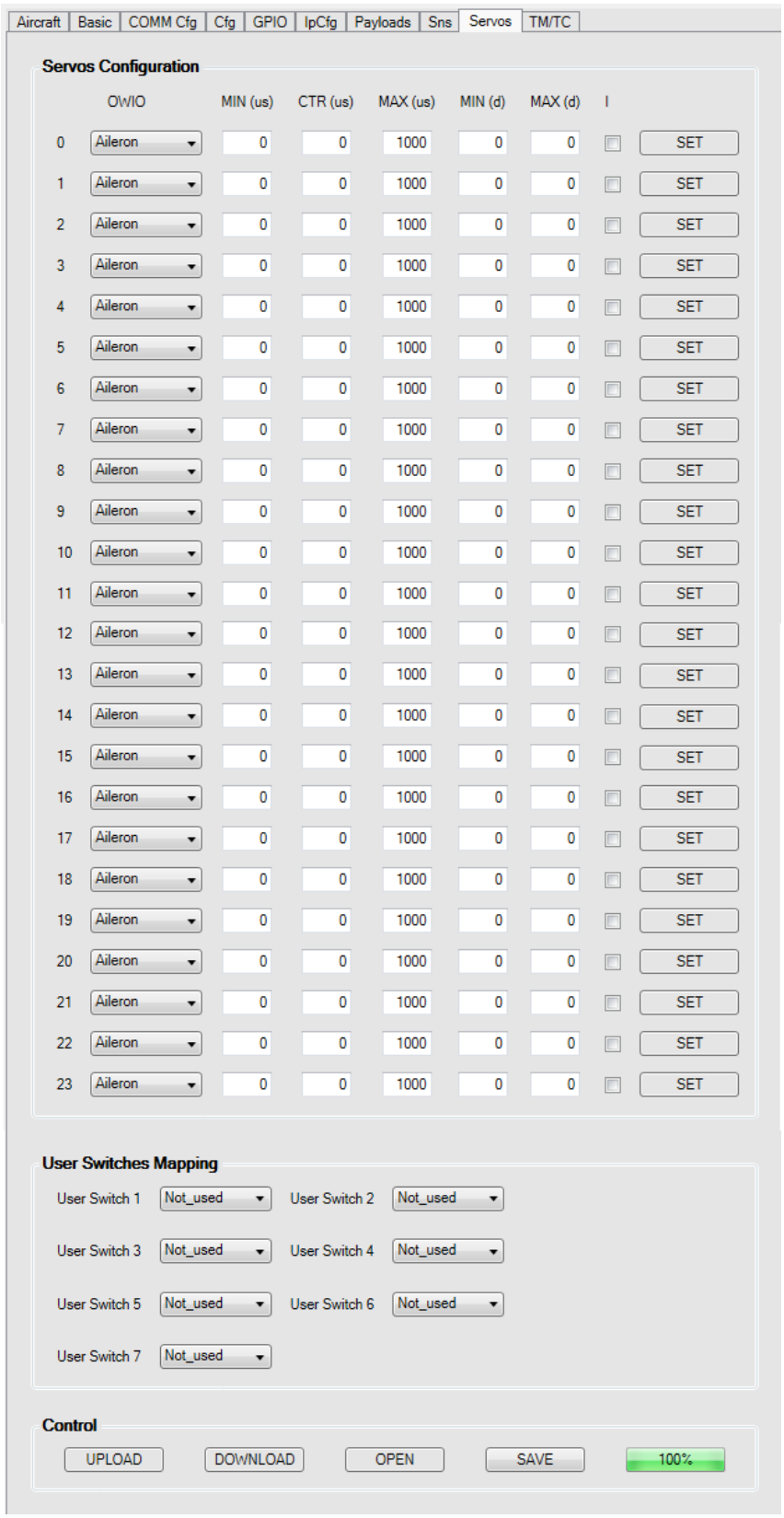


Figura F.10: Vista de la pestaña de configuración de servos para VECTOR.

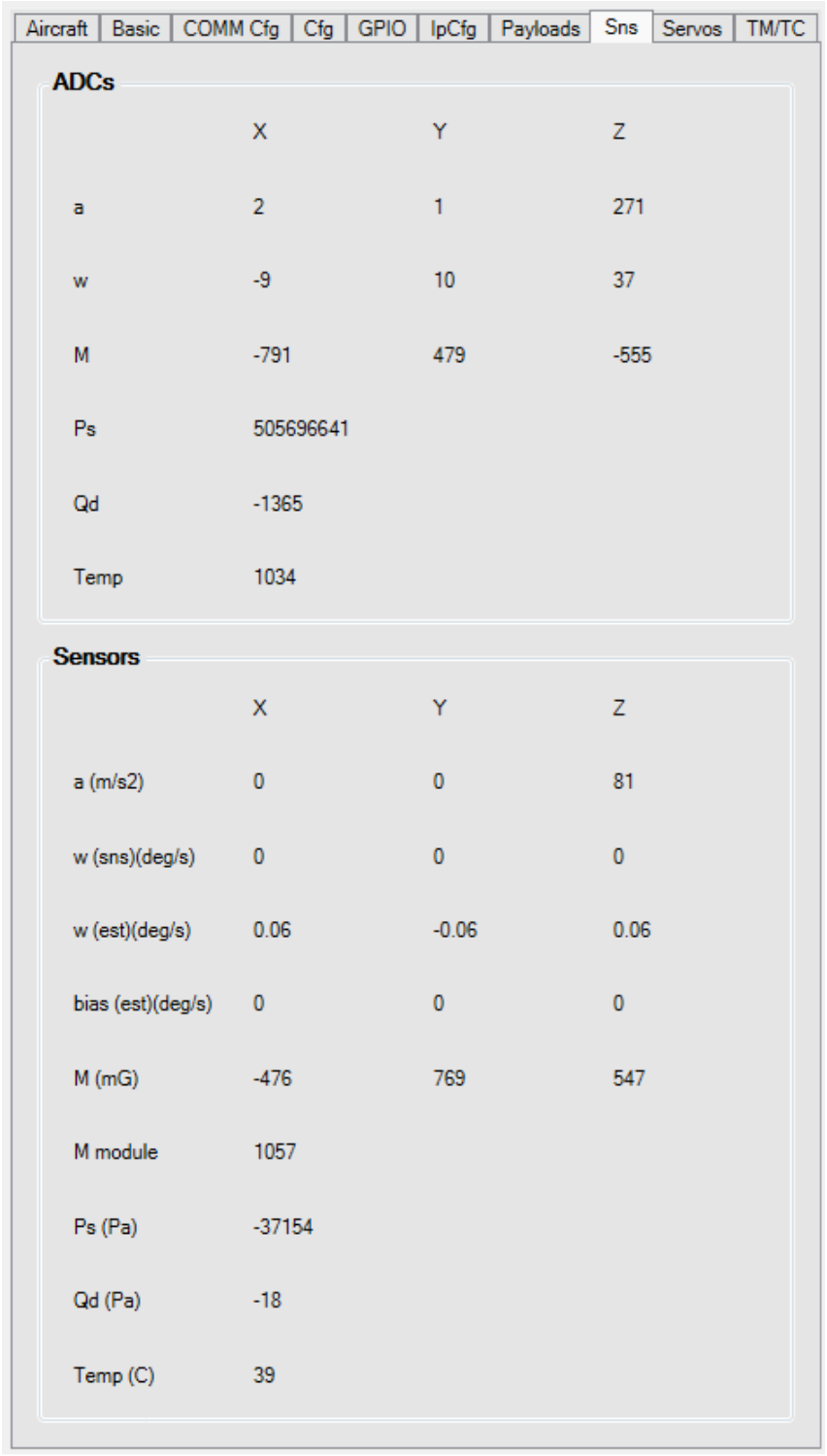


Figura F.11: Vista de la pestaña de sensores.

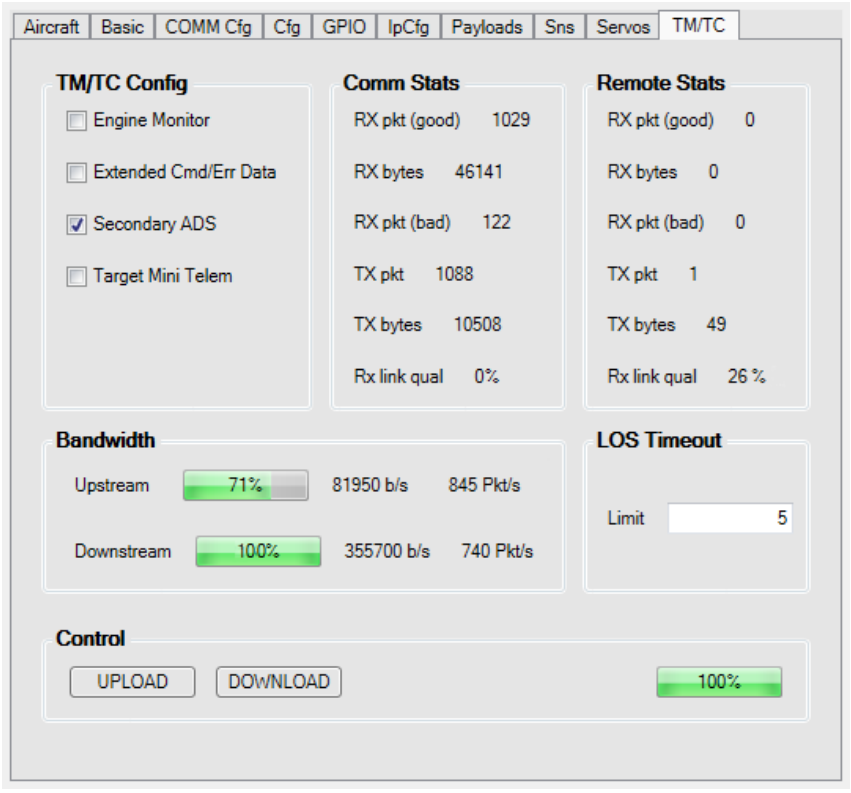


Figura F.12: Vista de la pestaña de telemetría.

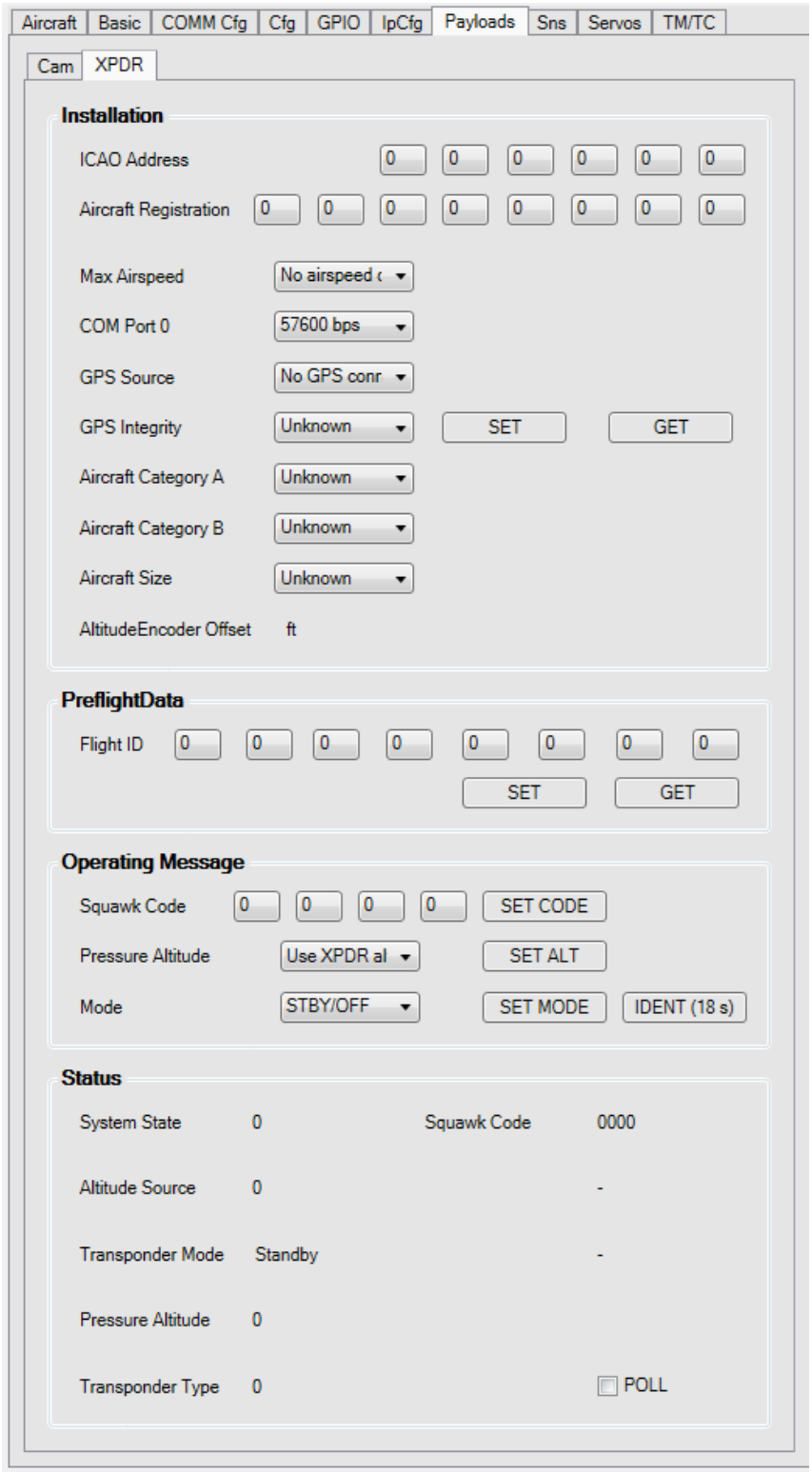


Figura F.13: Vista de la pestaña de configuración del transponedor.